

AD-A246 518


**SOFTWARE DESIGN DOCUMENT
Vehicle Simulation CSCI (5)**

Volume 1 of 4 Sections 1.0 - 2.2.3.1

June, 1991



Prepared by:

BBN Systems and Technologies,
A Division of Bolt Beranek and Newman Inc.
10 Moulton Street
Cambridge, MA 02138
(617) 873-3000 FAX: (617) 873-4315

Prepared for:

Defense Advanced Research Projects Agency (DARPA)
Information and Science Technology Office
1400 Wilson Blvd., Arlington, VA 22209-2308
(202) 694-8232, AUTOVON 224-8232

Program Manager for Training Devices (PM TRADE)
12350 Research Parkway
Orlando, FL 32826-3276
(407) 380-4518

This document has been approved
for public release and its
distribution is unlimited.

92 1 31 121


92-02595


Table of Contents

1	INTRODUCTION : VEHICLES CSCI	1
1.1	BACKGROUND.....	1
1.2	EXTERNAL INTERFACES.....	2
1.3	INTERNAL STRUCTURE.....	11
1.4	CONFIGURATION AND CONFIGURATION MANAGEMENT.....	12
1.5	TERMINOLOGY AND DOCUMENTATION.....	13
1.6	MISCELLANEOUS.....	13
2	CSC DESCRIPTIONS	14
2.1	DEVICE INTERFACE SOFTWARE.....	14
2.1.1	Network Interface Software	15
2.1.1.1	Network Device Interface	16
2.1.1.1.1	libnetif	16
2.1.1.2	Association Layer.....	16
2.1.1.2.1	libassoc	16
2.1.1.2.2	libp2p.....	17
2.1.1.2.2.1	init.c.....	17
2.1.1.2.2.1.1	PointToPointOpen.....	17
2.1.1.2.2.2	p2p.h.....	18
2.1.1.2.2.3	p2p_local.c	18
2.1.1.2.2.4	receive.c.....	19
2.1.1.2.2.4.1	PointToPointReceivePDU.....	19
2.1.1.2.2.5	send.c.....	20
2.1.1.2.2.5.1	PointToPointSendPDU	20
2.1.1.3	Host Network Interface	21
2.1.1.3.1	libSendNet.....	21
2.1.1.3.1.1	act_rsp.c.....	21
2.1.1.3.1.1.1	send_activate_response	22
2.1.1.3.1.2	activ_params.c.....	22
2.1.1.3.1.3	activate.c.....	22
2.1.1.3.1.4	appearance.c	23
2.1.1.3.1.4.1	format_vehicle_appearance.....	23
2.1.1.3.1.4.2	format_stealth_appearance.....	25
2.1.1.3.1.5	citv_instr.c	25
2.1.1.3.1.6	citv_orient.c.....	26
2.1.1.3.1.7	coll_rsp.c	26
2.1.1.3.1.7.1	network_send_collision_ response.....	26

Accession For	
NTIS GRA&I	J
DIC TAB	
Unannounced	
Justification	
By	
Distribution	
Approved for	
Dist	Special
A-1	



2.1.1.3.1.8	collision.c	27
2.1.1.3.1.8.1	network_send_outta_my_	
	way_mf.....	27
2.1.1.3.1.9	deact_rsp.c.....	28
2.1.1.3.1.9.1	network_send_deactivate_	
	response.....	28
2.1.1.3.1.10	deactivate.c.....	29
2.1.1.3.1.10.1	send_deactivate_pkt	29
2.1.1.3.1.11	death_status.c	30
2.1.1.3.1.11.1	network_set_death_status.....	30
2.1.1.3.1.11.2	network_set_smoking_status	30
2.1.1.3.1.11.3	network_set_burning_status.....	30
2.1.1.3.1.11.4	network_set_commo_kill	31
2.1.1.3.1.11.5	network_set_mobility_kill	31
2.1.1.3.1.11.6	network_set_firepower_kill	31
2.1.1.3.1.12	dust_status.c	32
2.1.1.3.1.12.1	network_set_dust_cloud.....	32
2.1.1.3.1.13	event_flag.c	32
2.1.1.3.1.13.1	network_send_event_flag.....	33
2.1.1.3.1.14	ex_status.c	34
2.1.1.3.1.14.1	send_exercise_status_pkt	34
2.1.1.3.1.14.2	send_exercise_status_trans.....	35
2.1.1.3.1.15	fuState.c.....	35
2.1.1.3.1.16	get_exer_id.c	35
2.1.1.3.1.16.1	network_get_exercise_id.....	35
2.1.1.3.1.17	get_force.c	36
2.1.1.3.1.17.1	network_get_vehicle_force	36
2.1.1.3.1.18	get_guises.c	36
2.1.1.3.1.18.1	network_get_vehicle_guises	36
2.1.1.3.1.19	get_sim_type.c.....	37
2.1.1.3.1.19.1	network_get_simulator_type.....	37
2.1.1.3.1.20	get_unit.c	37
2.1.1.3.1.20.1	network_get_vehicle_unit	37
2.1.1.3.1.21	get_veh_app.c.....	38
2.1.1.3.1.21.1	network_get_vehicle_	
	appearance.....	38
2.1.1.3.1.22	get_veh_id.c	38
2.1.1.3.1.22.1	network_get_vehicle_id	38
2.1.1.3.1.23	get_veh_type.c.....	39
2.1.1.3.1.23.1	network_get_vehicle_type	39
2.1.1.3.1.24	get_xmt_fail.c.....	39
2.1.1.3.1.24.1	net_xmt_failed.....	39
2.1.1.3.1.25	gnd_impact.c	40

2.1.1.3.1.25.1	network_send_ground_	
	impact	40
2.1.1.3.1.26	imp_rsp.c	41
2.1.1.3.1.26.1	network_send_impact_	
	response	41
2.1.1.3.1.27	laser_detect.c	41
2.1.1.3.1.28	laser_fire.c	41
2.1.1.3.1.29	laser_range.c	42
2.1.1.3.1.29.1	network_send_laser_range	42
2.1.1.3.1.30	laser_result.c	42
2.1.1.3.1.31	missile.c	43
2.1.1.3.1.31.1	network_missiles_init	43
2.1.1.3.1.31.2	network_send_missile_	
	appearance	44
2.1.1.3.1.31.3	network_stop_missile_flyout	45
2.1.1.3.1.31.4	network_send_missile_	
	fire_pkt	46
2.1.1.3.1.32	net_xmit.c	47
2.1.1.3.1.32.1	need_to_send_veh_status	47
2.1.1.3.1.32.2	network_xmit	47
2.1.1.3.1.32.3	network_xmit_idle	48
2.1.1.3.1.33	non_impact.c	49
2.1.1.3.1.33.1	network_send_non_impact	49
2.1.1.3.1.34	nprintf.c	50
2.1.1.3.1.34.1	nprintf	50
2.1.1.3.1.35	position.c	51
2.1.1.3.1.36	power_supply.c	51
2.1.1.3.1.37	reloadReq.c	51
2.1.1.3.1.38	repaired.c	51
2.1.1.3.1.38.1	send_repaired_pkt	51
2.1.1.3.1.39	resupp_cancel.c	52
2.1.1.3.1.40	resupp_offer.c	52
2.1.1.3.1.40.1	network_send_offer_packet	52
2.1.1.3.1.41	resupp_recvd.c	53
2.1.1.3.1.41.1	network_send_thank_you_	
	packet	53
2.1.1.3.1.42	send_dg_pkt.c	54
2.1.1.3.1.42.1	fill_simHdr	54
2.1.1.3.1.42.2	fill_mgmtHdr	54
2.1.1.3.1.42.3	fill_ivisHdr	54
2.1.1.3.1.42.4	fill_dcHdr	54
2.1.1.3.1.42.5	network_fill_hdr_send_	
	sim_pkt	55

2.1.1.3.1.42.6	network_fill_hdr_send_ dc_pkt	55
2.1.1.3.1.42.7	network_fill_hdr_send_ mgmt_pkt	56
2.1.1.3.1.42.8	network_fill_hdr_send_ ivis_pkt	56
2.1.1.3.1.43	send_loc.c	56
2.1.1.3.1.44	send_loc.h	57
2.1.1.3.1.45	send_pt_pkt.c	57
2.1.1.3.1.45.1	send_pt_packet	57
2.1.1.3.1.46	send_rsp.c	58
2.1.1.3.1.46.1	network_fill_hdr_send_ sim_rsp	58
2.1.1.3.1.46.2	network_fill_hdr_send_ dc_rsp	59
2.1.1.3.1.47	send_trans.c	60
2.1.1.3.1.47.1	network_fill_hdr_send_ sim_trans	60
2.1.1.3.1.47.2	network_fill_hdr_send_ dc_trans	61
2.1.1.3.1.48	service_req.c	62
2.1.1.3.1.48.1	network_send_feed_ me_packet	62
2.1.1.3.1.49	set_ex_id.c	63
2.1.1.3.1.49.1	network_set_exercise_id	63
2.1.1.3.1.50	set_force.c	63
2.1.1.3.1.50.1	network_set_force	63
2.1.1.3.1.51	set_guises.c	63
2.1.1.3.1.51.1	network_set_vehicle_guises	63
2.1.1.3.1.52	set_sim_type.c	64
2.1.1.3.1.52.1	network_set_simulator_type	64
2.1.1.3.1.53	set_veh_app.c	64
2.1.1.3.1.53.1	network_set_vehicle_ appearance	64
2.1.1.3.1.54	set_veh_clas.c	65
2.1.1.3.1.54.1	network_set_vehicle_class	65
2.1.1.3.1.55	set_veh_id.c	65
2.1.1.3.1.55.1	network_set_vehicle_id	65
2.1.1.3.1.56	set_xmt_fail.c	66
2.1.1.3.1.56.1	set_xmt_failed	66
2.1.1.3.1.57	shell_fire.c	67
2.1.1.3.1.57.1	network_send_shell_fire_pkt	67
2.1.1.3.1.58	show_effect.c	68
2.1.1.3.1.59	sim_status.c	68
2.1.1.3.1.59.1	send_simulation_status_pkt	68

2.1.1.3.1.59.2	send_simulation_status_trans.....	69
2.1.1.3.1.60	spec_appear.c	69
2.1.1.3.1.61	spec_status.c	69
2.1.1.3.1.62	stat_change.c	70
2.1.1.3.1.62.1	network_send_status_change.....	70
2.1.1.3.1.63	stat_rsp.c.....	71
2.1.1.3.1.63.1	network_respond_to_ query_trans	71
2.1.1.3.1.63.2	network_respond_to_ query_pkt.....	72
2.1.1.3.1.63.3	can_ignore	73
2.1.1.3.1.63.4	same_unit	74
2.1.1.3.1.63.5	included_unit.....	75
2.1.1.3.1.63.6	including_unit.....	76
2.1.1.3.1.63.7	send_status_response_trans.....	77
2.1.1.3.1.64	targetDiseng.c.....	77
2.1.1.3.1.65	target_engag.c	77
2.1.1.3.1.66	thresh.c	78
2.1.1.3.1.66.1	v_pkt_verbose_mode	78
2.1.1.3.1.66.2	network_stop_sending_app.....	78
2.1.1.3.1.66.3	network_restart_sending_app.....	78
2.1.1.3.1.66.4	network_check_veh_ appearance.....	79
2.1.1.3.1.66.5	network_init_thresholds.....	79
2.1.1.3.1.67	tow_status.c	80
2.1.1.3.1.67.1	network_tow_launcher_up.....	80
2.1.1.3.1.67.2	network_tow_launcher_down.....	80
2.1.1.3.1.68	use_activ.c	80
2.1.1.3.1.68.1	format_db_filename	80
2.1.1.3.1.68.2	network_use_activation.....	81
2.1.1.3.1.69	veh_app_loc.h	82
2.1.1.3.1.70	veh_impact.c	82
2.1.1.3.1.70.1	network_send_vehicle_ impact.....	83
2.1.1.3.1.71	veh_status.c	84
2.1.1.3.1.71.1	send_vehicle_status.....	84
2.1.1.3.1.71.2	send_vehicle_status_in_f____ ing_multicast_group_zero.....	85
2.1.1.3.1.71.3	send_vehicle_status_trans	85
2.1.1.3.1.71.4	build_vehicle_status.....	86
2.1.1.3.2	libRcvNet	87
2.1.1.3.2.1	activate.c.....	87
2.1.1.3.2.1.1	process_activate_request.....	87
2.1.1.3.2.2	alert_status.c.....	88

2.1.1.3.2.3	collision.c	88
2.1.1.3.2.3.1	process_collision	88
2.1.1.3.2.4	deactivate.c	89
2.1.1.3.2.4.1	process_deactivate_me	89
2.1.1.3.2.4.2	process_deactivate_other	89
2.1.1.3.2.5	fire.c	90
2.1.1.3.2.5.1	process_fire	90
2.1.1.3.2.6	fire_probe.c	90
2.1.1.3.2.7	idiot_check.c	91
2.1.1.3.2.8	impact.c	91
2.1.1.3.2.8.1	process_hit_me	91
2.1.1.3.2.8.2	process_hit_other	92
2.1.1.3.2.8.3	veh_impact_me	92
2.1.1.3.2.8.4	veh_impact_other	93
2.1.1.3.2.8.5	ground_impact	93
2.1.1.3.2.9	indir_fire.c	94
2.1.1.3.2.9.1	process_indirect_fire	94
2.1.1.3.2.10	laser_range.c	95
2.1.1.3.2.11	markers.c	95
2.1.1.3.2.11.1	process_markers	95
2.1.1.3.2.12	network_init.c	96
2.1.1.3.2.12.1	network_get_net_handle	96
2.1.1.3.2.12.2	network_set_net_layer	96
2.1.1.3.2.12.3	network_init	97
2.1.1.3.2.12.4	network_set_network_device	97
2.1.1.3.2.12.5	network_get_network_device	98
2.1.1.3.2.13	network_test.c	98
2.1.1.3.2.14	not_open_net.c	98
2.1.1.3.2.14.1	network_dont_really_open_ up_ethernet	98
2.1.1.3.2.15	open_net.c	98
2.1.1.3.2.15.1	network_really_open_up_ ethernet	98
2.1.1.3.2.16	print_stats.c	99
2.1.1.3.2.16.1	network_print_statistics	99
2.1.1.3.2.17	print_vimp.c	99
2.1.1.3.2.18	proc_a_pkt.c	99
2.1.1.3.2.18.1	set_process_pkt_fn	100
2.1.1.3.2.18.2	do_protocol_on_mgmt_ packet	100
2.1.1.3.2.18.3	do_protocol_on_data_ analysis_packet	100
2.1.1.3.2.18.4	do_protocol_on_sim_packet	101
2.1.1.3.2.18.5	process_sim_transaction	102

2.1.1.3.2.18.6	process_dc_transaction.....	102
2.1.1.3.2.18.7	reconstitute_from_keyboard.....	103
2.1.1.3.2.18.8	process_a_packet.....	103
2.1.1.3.2.19	prot_faad.c.....	104
2.1.1.3.2.20	prot_ivis.c.....	104
2.1.1.3.2.21	prot_laser.c.....	104
2.1.1.3.2.22	prot_stealth.c.....	104
2.1.1.3.2.23	radiate.c.....	104
2.1.1.3.2.24	rad_state.c.....	104
2.1.1.3.2.24.1	process_radiating_state.....	104
2.1.1.3.2.25	rcv_loc.c.....	105
2.1.1.3.2.26	rcv_loc.h.....	105
2.1.1.3.2.27	really.c.....	105
2.1.1.3.2.27.1	network_can_i_really_ use_network.....	105
2.1.1.3.2.28	repair.c.....	106
2.1.1.3.2.28.1	process_repair.....	106
2.1.1.3.2.29	resupp_canc.c.....	107
2.1.1.3.2.29.1	process_resupply_cancel.....	107
2.1.1.3.2.30	resupp_offer.c.....	108
2.1.1.3.2.30.1	process_resupply_offer.....	108
2.1.1.3.2.31	resupp_recvd.c.....	109
2.1.1.3.2.31.1	process_resupply_received.....	109
2.1.1.3.2.32	service_req.c.....	110
2.1.1.3.2.32.1	process_service_request.....	110
2.1.1.3.2.33	show_effect.c.....	110
2.1.1.3.2.34	status_query.c.....	111
2.1.1.3.2.34.1	process_status_query.....	111
2.1.1.3.2.34.2	process_query_me.....	111
2.1.1.3.2.35	tgt_fire_cmd.c.....	111
2.1.1.3.2.36	tgt_handoff.c.....	112
2.1.1.3.2.37	tgt_vis.c.....	112
2.1.1.3.2.38	veh_appear.c.....	112
2.1.1.3.2.38.1	process_update.....	112
2.1.2	CIG Interface Software.....	113
2.1.2.1	CIG Device Interface.....	114
2.1.2.1.1	libcif.....	114
2.1.2.1.1.1	connect.c.....	114
2.1.2.1.1.1.1	cif_connect.....	114
2.1.2.1.1.2	data.c.....	116
2.1.2.1.1.3	disconnect.c.....	116
2.1.2.1.1.3.1	cif_disconnect.....	116
2.1.2.1.1.4	init.c.....	118

2.1.2.1.1.4.1	cif_init	118
2.1.2.1.1.5	libcif.h	119
2.1.2.1.1.6	parse.c	120
2.1.2.1.1.6.1	parse_cif_definition.....	120
2.1.2.1.1.7	receive.c.....	121
2.1.2.1.1.7.1	cif_receive	121
2.1.2.1.1.7.2	dr11_receive	123
2.1.2.1.1.8	send.c.....	124
2.1.2.1.1.8.1	cif_send	124
2.1.2.1.1.8.2	dr11_send	126
2.1.2.1.1.9	uninit.c.....	127
2.1.2.1.1.9.1	cif_uninit	127
2.1.2.2	CIG-SIM Buffer Interface.....	128
2.1.2.2.1	libcig.....	128
2.1.2.2.1.1	check_sizes.c	128
2.1.2.2.1.1.1	check_buffer_sizes	129
2.1.2.2.1.2	cig_local.c	129
2.1.2.2.1.3	cig_local.h	129
2.1.2.2.1.4	cig_no_op.c	130
2.1.2.2.1.4.1	cig_prepare_no_op	130
2.1.2.2.1.5	cig_nuse_gra.c.....	130
2.1.2.2.1.5.1	cig_not_using_graphics.....	130
2.1.2.2.1.6	cig_prepare.c	131
2.1.2.2.1.6.1	cig_prepare	131
2.1.2.2.1.7	cig_proc_buf.c.....	132
2.1.2.2.1.7.1	cig_process_buffer	132
2.1.2.2.1.8	cig_r_start.c.....	132
2.1.2.2.1.8.1	use_print_checkb.....	133
2.1.2.2.1.8.2	set_ded_name	133
2.1.2.2.1.8.3	cig_reconfig_start.....	133
2.1.2.2.1.9	cig_recv_buf.c.....	133
2.1.2.2.1.9.1	cig_receive_buffer.....	134
2.1.2.2.1.10	cig_send_buf.c.....	135
2.1.2.2.1.10.1	cig_send_buffer.....	136
2.1.2.2.1.10.2	cig_kickoff_dr_xfer.....	137
2.1.2.2.1.10.3	cig_poll_dr_transfer	137
2.1.2.2.1.10.4	cig_setup_dr_transfer	137
2.1.2.2.1.11	cig_set_conf.c.....	138
2.1.2.2.1.11.1	cig_setup_configuration	138
2.1.2.2.1.12	cig_stop.c.....	139
2.1.2.2.1.12.1	cig_stop	139
2.1.2.2.1.13	cig_sync.c.....	140
2.1.2.2.1.13.1	cig_synchronize.....	140

2.1.2.2.1.14	cig_uninit.c.....	141
2.1.2.2.1.14.1	cig_uninit.....	141
2.1.2.2.1.15	cig_use_gra.c.....	141
2.1.2.2.1.15.1	cig_using_graphics.....	141
2.1.2.2.1.16	db_override.c.....	142
2.1.2.2.1.16.1	cig_use_database_ override_named.....	142
2.1.2.2.1.17	get_cig2.c	142
2.1.2.2.1.17	get_cig2_present.....	142
2.1.2.2.1.18	get_i_sizes.c	143
2.1.2.2.1.18.1	get_initial_sizes.....	143
2.1.2.2.1.19	get_max.c	143
2.1.2.2.1.19.1	get_max_buffer_sizes.....	143
2.1.2.2.1.20	get_r_size.c.....	144
2.1.2.2.1.20.1	get_receive_size	144
2.1.2.2.1.21	get_rcv_buf.c	144
2.1.2.2.1.21.1	get_receive_buffer.....	144
2.1.2.2.1.22	get_s_size.c	145
2.1.2.2.1.22.1	get_send_size	145
2.1.2.2.1.23	not_prep_buf.c.....	145
2.1.2.2.1.23.1	cig_not_ok_to_prepare_ buffer	145
2.1.2.2.1.24	not_proc_buf.c.....	146
2.1.2.2.1.24.1	cig_not_ok_to_process_ buffer	146
2.1.2.2.1.25	send_status.c.....	146
2.1.2.2.1.25.1	get_send_status.....	146
2.1.2.2.1.25.2	set_send_status.....	147
2.1.2.2.1.26	set_cig_dev.c	147
2.1.2.2.1.26.1	set_cig_dev.....	147
2.1.2.2.1.27	set_i_sizes.c.....	148
2.1.2.2.1.27.1	set_initial_sizes	148
2.1.2.2.1.28	set_my_if.c	148
2.1.2.2.1.28.1	set_my_if.....	148
2.1.2.2.1.29	set_req_rcv.c.....	149
2.1.2.2.1.29.1	set_request_receive_size	149
2.1.2.2.1.30	set_req_send.c	149
2.1.2.2.1.30.1	set_request_send_size	149
2.1.2.2.1.31	set_s_flag.c.....	150
2.1.2.2.1.31.1	set_use_requested_flag.....	150
2.1.2.2.1.32	setup_buf.c	150
2.1.2.2.1.32.1	setup_buffer_ptrs.....	151
2.1.2.2.1.33	cig_get_db.c	151

2.1.2.2.1.33.1	cig_get_db	151
2.1.2.2.2	libmsg	152
2.1.2.2.2.1	add_veh2cig.c	152
2.1.2.2.2.1.1	add_veh_to_cig_msg	153
2.1.2.2.2.2	adj_chg_stat.c	154
2.1.2.2.2.2.1	fill_changed_static_remove_ msg	154
2.1.2.2.2.2.2	fill_changed_static_msg	155
2.1.2.2.2.2.3	cig_adjust_for_changed_ staticveh	156
2.1.2.2.2.3	adj_otherveh.c	157
2.1.2.2.2.3.1	cig_msg_adjust_otherveh_ state	157
2.1.2.2.2.4	app_end.c	158
2.1.2.2.2.4.1	cig_msg_append_end	158
2.1.2.2.2.5	app_msg_hdr.c	159
2.1.2.2.2.5.1	append_msg_hdr	159
2.1.2.2.2.6	app_mtra_ent.c	160
2.1.2.2.2.6.1	multi_cig_append_traj_ entry_xfer	160
2.1.2.2.2.7	app_mtra_tbl.c	161
2.1.2.2.2.7.1	multi_cig_append_ traj_table_xfer	161
2.1.2.2.2.8	app_stat_rm.c	162
2.1.2.2.2.8.1	cig_msg_append_staticveh_ rem	162
2.1.2.2.2.9	app_stat_veh.c	163
2.1.2.2.2.9.1	cig_msg_append_staticveh_ state	163
2.1.2.2.2.10	app_traj_ent.c	164
2.1.2.2.2.10.1	cig_msg_append_traj_ entry_xfer	164
2.1.2.2.2.11	app_traj_tbl.c	165
2.1.2.2.2.11.1	cig_msg_append_traj_ table_xfer	165
2.1.2.2.2.12	app_vflags.c	166
2.1.2.2.2.12.1	cig_msg_append_view_flags	166
2.1.2.2.2.13	append_other.c	167
2.1.2.2.2.13.1	append_other_in_send_buffer	167
2.1.2.2.2.14	ball_buffer.c	167
2.1.2.2.2.14.1	init_ballistics_buffer	168
2.1.2.2.2.14.2	copy_ballistics_buffer	168
2.1.2.2.2.14.3	store_traj_chord	169
2.1.2.2.2.14.4	store_round_fired	170
2.1.2.2.2.14.5	store_view_magnification	171

2.1.2.2.2.14.6	store_other_veh_state.....	171
2.1.2.2.2.14.7	store_ctas_init_startup_model.....	171
2.1.2.2.2.14.8	store_ctas_grow_model.....	171
2.1.2.2.2.15	buf_reset.c	172
2.1.2.2.2.15.1	buffer_reset.....	172
2.1.2.2.2.16	buf_setup.c	172
2.1.2.2.2.16.1	buffer_setup.....	172
2.1.2.2.2.17	check_all.c.....	173
2.1.2.2.2.17.1	check_all.....	173
2.1.2.2.2.18	checkbuffer.c	174
2.1.2.2.2.18.1	check_buffer.....	174
2.1.2.2.2.18.2	print_msg_pass_on.....	175
2.1.2.2.2.19	cig_flushbuf.c.....	175
2.1.2.2.2.19.1	cig_flush_buffer	175
2.1.2.2.2.20	clr_n_mapped.c	175
2.1.2.2.2.20.1	clear_n_mapped	175
2.1.2.2.2.21	config_key.c	176
2.1.2.2.2.21.1	key_list_initialized	176
2.1.2.2.2.21.2	key_list_init.....	177
2.1.2.2.2.21.3	add_keyword	177
2.1.2.2.2.21.4	lookup_keyword.....	178
2.1.2.2.2.22	config_key.h.....	178
2.1.2.2.2.23	config_msg.c	179
2.1.2.2.2.23.1	cig_set_view_config_file	179
2.1.2.2.2.23.2	cig_set_traj_config_file.....	180
2.1.2.2.2.23.3	cig_msg_configure_traj.....	180
2.1.2.2.2.23.4	cig_msg_configure_view	180
2.1.2.2.2.24	config_read.c	181
2.1.2.2.2.24.1	config_pos_init.....	181
2.1.2.2.2.24.2	config_pos_init2.....	182
2.1.2.2.2.24.3	cig_read_configfile.....	183
2.1.2.2.2.24.4	read_keyword_data	184
2.1.2.2.2.24.5	process_keyword.....	184
2.1.2.2.2.24.6	send_buffer.....	185
2.1.2.2.2.25	dealloc_abuf.c	185
2.1.2.2.2.25.1	deallocate_appended_buffer_ space	185
2.1.2.2.2.26	dealloc_pbuf.c	186
2.1.2.2.2.26.1	deallocate_prepended_buffer_ space	186
2.1.2.2.2.27	del_veh.c	187
2.1.2.2.2.27.1	delete_veh_from_cig_msg	187
2.1.2.2.2.28	flushbuf.c.....	188

2.1.2.2.2.28.1	flush_buffer	188
2.1.2.2.2.29	get_back.c	188
2.1.2.2.2.29.1	get_back_of_send_buffer	188
2.1.2.2.2.30	get_cig_mask.c	189
2.1.2.2.2.30.1	get_cig_mask	189
2.1.2.2.2.31	get_debug.c	189
2.1.2.2.2.31.1	get_static_debug	189
2.1.2.2.2.32	get_front.c	190
2.1.2.2.2.32.1	get_front_of_send_buffer	190
2.1.2.2.2.33	get_init_buf.c	190
2.1.2.2.2.33.1	get_init_ptrs	190
2.1.2.2.2.34	get_n_mapped.c	191
2.1.2.2.2.34.1	get_n_mapped	191
2.1.2.2.2.35	get_other_st.c	191
2.1.2.2.2.35.1	get_other_start_in_send_ buffer	191
2.1.2.2.2.36	get_sbuffer.c	192
2.1.2.2.2.36.1	get_sbuffer	192
2.1.2.2.2.37	msg_loc.c	192
2.1.2.2.2.38	msg_loc.h	193
2.1.2.2.2.40	pr_agl.c	194
2.1.2.2.2.40.1	print_msg_agl	194
2.1.2.2.2.41	pr_cig_ctl.c	194
2.1.2.2.2.41.1	print_msg_cig_ctl	194
2.1.2.2.2.42	pr_ct_gm.c	195
2.1.2.2.2.43	pr_ct_ism.c	195
2.1.2.2.2.44	pr_end.c	195
2.1.2.2.2.44.1	print_msg_end	195
2.1.2.2.2.45	pr_file_desc.c	195
2.1.2.2.2.45.1	print_msg_file_descr	195
2.1.2.2.2.46	pr_file_stat.c	196
2.1.2.2.2.46.1	print_msg_file_status	196
2.1.2.2.2.47	pr_file_xfer.c	196
2.1.2.2.2.47.1	print_msg_file_xfer	196
2.1.2.2.2.48	pr_hit.c	197
2.1.2.2.2.48.1	print_msg_hit	197
2.1.2.2.2.49	pr_hit_rtn.c	198
2.1.2.2.2.49.1	print_msg_hit_return	198
2.1.2.2.2.50	pr_laser_rtn.c	198
2.1.2.2.2.50.1	print_msg_laser_return	198
2.1.2.2.2.51	pr_loc_terr.c	199
2.1.2.2.2.51.1	print_msg_local_terrain	199
2.1.2.2.2.52	pr_miss.c	200

2.1.2.2.2.52.1	print_msg_miss	200
2.1.2.2.2.53	pr_otherveh.c	201
2.1.2.2.2.53.1	print_msg_otherveh_state	201
2.1.2.2.2.54	pr_proc_rnd.c	202
2.1.2.2.2.54.1	print_msg_process_round	202
2.1.2.2.2.55	pr_rnd_fired.c	203
2.1.2.2.2.55.1	print_msg_round_fired	203
2.1.2.2.2.56	pr_show_eff.c	204
2.1.2.2.2.56.1	print_msg_show_effect	204
2.1.2.2.2.57	pr_staticrem.c	205
2.1.2.2.2.57.1	print_msg_staticveh_rem	205
2.1.2.2.2.58	pr_staticveh.c	206
2.1.2.2.2.58.1	print_msg_staticveh_state	206
2.1.2.2.2.59	pr_submode.c	207
2.1.2.2.2.59.1	print_msg_subsys_mode	207
2.1.2.2.2.60	pr_sys_err.c	207
2.1.2.2.2.60.1	print_msg_sys_error	207
2.1.2.2.2.61	pr_test_name.c	208
2.1.2.2.2.61.1	print_msg_test_name	208
2.1.2.2.2.62	pr_traj_chrd.c	209
2.1.2.2.2.62.1	print_msg_traj_chord	209
2.1.2.2.2.63	pr_vupdate.c	209
2.1.2.2.2.64	pre_1rot.c	210
2.1.2.2.2.64.1	cig_msg_prepend_1rotation	210
2.1.2.2.2.65	pre_3rot.c	211
2.1.2.2.2.65.1	cig_msg_prepend_3rotations	211
2.1.2.2.2.66	pre_agl_set.c	212
2.1.2.2.2.66.1	cig_msg_prepend_agl_setup	212
2.1.2.2.2.67	pre_am_dfn.c	213
2.1.2.2.2.67.1	cig_msg_prepend_ammo_ define	213
2.1.2.2.2.68	pre_cig_ctl.c	214
2.1.2.2.2.68.1	cig_msg_prepend_cig_ctl	214
2.1.2.2.2.69	pre_ct_gm.c	214
2.1.2.2.2.70	pre_ct_ism.c	214
2.1.2.2.2.71	pre_dr11.c	215
2.1.2.2.2.71.1	cig_msg_prepend_dr11_pkt_ size	215
2.1.2.2.2.72	pre_file_des.c	216
2.1.2.2.2.72.1	push_msg_file_descr	216
2.1.2.2.2.73	pre_file_sts.c	217
2.1.2.2.2.73.1	push_msg_file_status	217
2.1.2.2.2.74	pre_file_xfr.c	218

2.1.2.2.2.74.1	push_msg_file_xfer.....	218
2.1.2.2.2.75	pre_gun_over.c.....	219
2.1.2.2.2.75.1	cig_msg_prepend_gun_ overlay.....	219
2.1.2.2.2.76	pre_hprxyzs.c.....	220
2.1.2.2.2.76.1	cig_msg_prepend_hprxyzs_ matrix.....	220
2.1.2.2.2.77	pre_lase_rtn.c.....	221
2.1.2.2.2.77.1	push_msg_laser_return.....	221
2.1.2.2.2.78	pre_mcig_ctl.c.....	222
2.1.2.2.2.78.1	multi_cig_push_cig_ctl.....	222
2.1.2.2.2.79	pre_mdr11.c.....	223
2.1.2.2.2.79.1	multi_cig_msg_prepend_ dr11_ pkt_size.....	223
2.1.2.2.2.80	pre_mpass_on.c.....	224
2.1.2.2.2.80.1	multi_cig_msg_prepend_ pass_on.....	224
2.1.2.2.2.81	pre_mreq_1sr.c.....	225
2.1.2.2.2.81.1	multi_cig_msg_prepend_ request_laser_range.....	225
2.1.2.2.2.82	pre_mrts4x3.c.....	226
2.1.2.2.2.82.1	multi_cig_msg_prepend_ rts4x3_matrix.....	226
2.1.2.2.2.83	pre_msg_hdr.c.....	227
2.1.2.2.2.83.1	prepend_msg_hdr.....	227
2.1.2.2.2.84	pre_mvflags.c.....	228
2.1.2.2.2.84.1	multi_cig_msg_prepend_ view_flags.....	228
2.1.2.2.2.85	pre_obscure.c.....	229
2.1.2.2.2.85.1	cig_msg_prepend_obscure.....	229
2.1.2.2.2.86	pre_overall.c.....	230
2.1.2.2.2.86.1	cig_msg_prepend_overall_ hdr.....	230
2.1.2.2.2.87	pre_ovr_set.c.....	231
2.1.2.2.2.87.1	cig_msg_prepend_overlay_ setup.....	231
2.1.2.2.2.88	pre_pass_bk.c.....	232
2.1.2.2.2.88.1	cig_msg_prepend_pass_bk.....	232
2.1.2.2.2.89	pre_pass_on.c.....	233
2.1.2.2.2.89.1	cig_msg_prepend_pass_on.....	233
2.1.2.2.2.90	pre_proc_md.c.....	233
2.1.2.2.2.90.1	cig_msg_prepend_ballistics_ msg.....	234
2.1.2.2.2.91	pre_req_1sr.c.....	235

2.1.2.2.2.91.1	cig_msg_prepend_request_ laser_range	235
2.1.2.2.2.92	pre_md_fir.c	235
2.1.2.2.2.92.1	cig_msg_prepend_ballistics_ msg	236
2.1.2.2.2.93	pre_rot2x1.c	237
2.1.2.2.2.93.1	cig_msg_prepend_rot2x1_ matrix	237
2.1.2.2.2.94	pre_rts4x3.c	238
2.1.2.2.2.94.1	cig_msg_prepend_rts4x3_ matrix	238
2.1.2.2.2.95	pre_scale.c	239
2.1.2.2.2.95.1	cig_msg_prepend_scale	239
2.1.2.2.2.96	pre_show_eff.c	240
2.1.2.2.2.96.1	cig_msg_prepend_show_ effect	240
2.1.2.2.2.97	pre_stat_rm.c	241
2.1.2.2.2.97.1	cig_msg_prepend_staticveh_ rem	241
2.1.2.2.2.98	pre_stat_veh.c	242
2.1.2.2.2.98.1	cig_msg_prepend_staticveh_ state	242
2.1.2.2.2.99	pre_submode.c	243
2.1.2.2.2.100	pre_sys_err.c	243
2.1.2.2.2.100.1	cig_msg_prepend_sys_error	243
2.1.2.2.2.101	pre_test_nam.c	244
2.1.2.2.2.101.1	cig_msg_prepend_test_name	244
2.1.2.2.2.102	pre_traj_chd.c	245
2.1.2.2.2.102.1	cig_msg_prepend_traj_chord	245
2.1.2.2.2.103	pre_traj_ent.c	246
2.1.2.2.2.103.1	cig_msg_prepend_traj_entry_ xfer	246
2.1.2.2.2.104	pre_traj_tbl.c	247
2.1.2.2.2.104.1	cig_msg_prepend_traj_table_ xfer	247
2.1.2.2.2.105	pre_trans.c	248
2.1.2.2.2.105.1	cig_msg_prepend_update_ translation	248
2.1.2.2.2.106	pre_vflags.c	249
2.1.2.2.2.106.1	cig_msg_prepend_view_flags	249
2.1.2.2.2.107	pre_vmag.c	250
2.1.2.2.2.107.1	cig_msg_prepend_view_ magnification	250
2.1.2.2.2.108	pre_vmode.c	251

2.1.2.2.2.108.1	cig_msg_prepend_view_ mode	251
2.1.2.2.2.109	pre_vport.c	252
2.1.2.2.2.109.1	cig_msg_prepend_viewport_ state	252
2.1.2.2.2.110	pre_vupdate.c	253
2.1.2.2.2.111	printbuffer.c	253
2.1.2.2.2.111.1	print_buffer	253
2.1.2.2.2.112	set_assym.c	254
2.1.2.2.2.112.1	set_assymmetric_on	254
2.1.2.2.2.113	set_buf_num.c	254
2.1.2.2.2.113.1	set_buffer_num	254
2.1.2.2.2.114	set_cig_mask.c	255
2.1.2.2.2.114.1	set_cig_mask	255
2.1.2.2.2.115	set_veh_spec.c	255
2.1.2.2.2.115.1	cig_set_veh_spec_ptrs	255
2.1.2.2.2.116	use_debug.c	256
2.1.2.2.2.116.1	use_static_debug	256
2.1.2.2.2.117	libmsg.h	256
2.1.2.2.3	libproc	259
2.1.2.2.3.1	alt_abv_gnd.c	259
2.1.2.2.3.1.1	cig_altitude_above_gnd	259
2.1.2.2.3.2	get_f_status.c	260
2.1.2.2.3.2.1	cig_get_file_status_data	260
2.1.2.2.3.3	get_file_dat.c	261
2.1.2.2.3.3.1	cig_get_file_xfer_data	261
2.1.2.2.3.4	get_laser.c	261
2.1.2.2.3.4.1	cig_laser_range	261
2.1.2.2.3.5	get_laser2.c	262
2.1.2.2.3.5.1	cig_laser_range2	262
2.1.2.2.3.6	init_agl_rtn.c	262
2.1.2.2.3.6.1	cig_init_msg_agl_routine	262
2.1.2.2.3.7	proc_agl.c	263
2.1.2.2.3.7.1	process_msg_agl	263
2.1.2.2.3.8	proc_buf.c	263
2.1.2.2.3.8.1	process_buffer	264
2.1.2.2.3.9	proc_ct_ram.c	264
2.1.2.2.3.10	proc_end.c	265
2.1.2.2.3.11	proc_f_stat.c	265
2.1.2.2.3.11.1	process_msg_file_status	265
2.1.2.2.3.12	proc_f_xfer.c	266
2.1.2.2.3.12.1	process_msg_file_xfer	266
2.1.2.2.3.13	proc_fdescr.c	267

2.1.2.2.3.13.1	process_msg_file_descr	267
2.1.2.2.3.14	proc_hit.c	267
2.1.2.2.3.14.1	process_msg_hit_return	268
2.1.2.2.3.14.2	process_msg_hit	269
2.1.2.2.3.15	proc_l_terr.c	269
2.1.2.2.3.15.1	process_msg_local_terrain	269
2.1.2.2.3.16	proc_laser.c	270
2.1.2.2.3.16.1	process_msg_laser_return	270
2.1.2.2.3.17	proc_loc.c	270
2.1.2.2.3.18	proc_loc.h	271
2.1.2.2.3.19	proc_lt_pi.c	271
2.1.2.2.3.19.1	process_msg_lt_piece	271
2.1.2.2.3.20	proc_miss.c	272
2.1.2.2.3.20.1	process_msg_miss	272
2.1.2.2.3.21	proc_pback.c	272
2.1.2.2.3.22	proc_sys_err.c	273
2.1.2.2.3.22.1	process_msg_sys_error	273
2.1.2.2.3.23	set_chunk.c	274
2.1.2.2.3.23.1	set_chunk_size	274
2.1.2.2.4	libvflags	275
2.1.2.2.4.1	clr_br_bit.c	275
2.1.2.2.4.1.1	clr_br_bit	275
2.1.2.2.4.2	clr_vflags.c	276
2.1.2.2.4.2.1	clear_view_flags	276
2.1.2.2.4.3	get_br_vals.c	277
2.1.2.2.4.3.1	get_br_vals	277
2.1.2.2.4.4	get_vflags.c	278
2.1.2.2.4.4.1	get_view_flags	278
2.1.2.2.4.5	get_vmodes.c	279
2.1.2.2.4.5.1	get_vmodes	279
2.1.2.2.4.6	set_br_bit.c	280
2.1.2.2.4.6.1	set_br_bit	280
2.1.2.2.4.7	set_br_vals.c	280
2.1.2.2.4.7.1	set_br_vals	280
2.1.2.2.4.8	set_vflags.c	281
2.1.2.2.4.8.1	set_view_flags	281
2.1.2.2.4.9	set_vmodes.c	281
2.1.2.2.4.9.1	set_vmodes	281
2.1.2.2.4.10	vflags_loc.c	282
2.1.2.2.4.11	vision.c	282
2.1.2.2.4.12	vflags_loc.h	282
2.1.2.2.4.13	libvflags.h	282
2.1.2.2.5	libio_simul	283

2.1.2.2.5.1	io_simul.c	283
2.1.2.2.5.1.1	io_simul	284
2.1.2.2.5.1.2	io_simul_idle	285
2.1.2.2.6	m1_cig.c	286
2.1.2.2.6.1	set_ballistics_debug	287
2.1.2.2.6.2	get_ballistics_debug	287
2.1.2.2.6.3	cig_init_ctr	287
2.1.2.2.6.4	cig_gps_mag_10x	287
2.1.2.2.6.4	cig_gps_mag_3x	288
2.1.2.2.6.5	cig_msg_prepend_my_veh_ state	288
2.1.2.2.6.6	cig_prepare_buffer	288
2.1.2.2.6.7	cig_spec_init	290
2.1.2.2.6.8	cig_setup_configuration	290
2.1.2.2.7	m2_cig.c	291
2.1.2.2.7.1	set_ballistics_debug	292
2.1.2.2.7.2	get_ballistics_debug	292
2.1.2.2.7.3	init_brow_pad_state	292
2.1.2.2.7.4	cig_msg_prepend_my_veh_ state	293
2.1.2.2.7.5	cig_prepare_buffer	293
2.1.2.2.7.6	cig_spec_init	295
2.1.2.2.8	kato_cig.c	296
2.1.2.2.8.1	cig_init_ctr	296
2.1.2.2.8.2	cig_local_init	296
2.1.2.2.8.3	cig_msg_prepend_my_veh_ state	297
2.1.2.2.8.4	cig_prepare_buffer	298
2.1.2.2.8.5	cig_spec_init	299
2.1.2.2.9	kato_view.c	300
2.1.2.2.9.1	view	301
2.1.2.2.9.2	view_init	301
2.1.2.2.9.3	view_simul	302
2.1.2.2.9.4	view_set_cpo_elevate_rate	302
2.1.2.2.9.5	view_set_pitch_rate	302
2.1.2.2.9.6	view_centered	303
2.1.2.2.9.7	view_up_depressed	303
2.1.2.2.9.8	view_up_released	303
2.1.2.2.9.9	view_down_depressed	303
2.1.2.2.9.10	view_down_released	303
2.1.2.2.9.11	view_to_world	304
2.1.2.2.9.12	view_get_desired_missile_ heading	304
2.1.2.2.9.13	view_get_pitch_angle	304

	2.1.2.2.9.14	view_get_yaw_angle.....	305
	2.1.2.2.9.15	yaw_filter	305
	2.1.2.2.9.16	pitch_filter	306
	2.1.2.2.9.17	view_set_pitch_angle.....	306
2.1.3.1	libsound.c		307
	2.1.3.1.1	sound_of_weapons_impact	308
	2.1.3.1.2	sound_make_const_sound.....	309
	2.1.3.1.3	sound_force_const_sound	309
	2.1.3.1.4	sound_make_var_sound.....	310
	2.1.3.1.5	sound_get_var_sound_arg.....	310
	2.1.3.1.6	sound_make_arg_sound.....	311
	2.1.3.1.7	sound_make_del_sound	311
	2.1.3.1.8	sound_force_del_sound.....	311
	2.1.3.1.9	sound_make_cont_sound	312
	2.1.3.1.10	sound_stop_cont_sound	313
2.1.3.2	m1_sound.c		314
	2.1.3.2.1	sound_denial_check	315
	2.1.3.2.2	sound_make_veh_spec_ sound	315
	2.1.3.2.3	sound_force_veh_spec_sound.....	316
	2.1.3.2.4	sound_init.....	316
	2.1.3.2.5	sound_dont_use.....	316
	2.1.3.2.6	sound_simul	316
	2.1.3.2.7	sound reset.....	317
	2.1.3.2.8	sound_we_just_died.....	317
	2.1.3.2.9	sound_of_tracks	317
	2.1.3.2.10	sound_of_turret_traversing	318
	2.1.3.2.11	sound_of_gun_elevating	318
	2.1.3.2.12	sound_of_random_sounds.....	319
2.1.3.3	m2_sound.c		320
	2.1.3.3.1	sound_denial_check	321
	2.1.3.3.2	sound_make_veh_spec_ sound	322
	2.1.3.3.3	sound_force_veh_spec_sound.....	322
	2.1.3.3.4	sound_init.....	322
	2.1.3.3.5	sound_dont_use.....	323
	2.1.3.3.6	sound_simul	323
	2.1.3.3.7	sound reset.....	323
	2.1.3.3.8	sound_we_just_died.....	323
	2.1.3.3.9	sound_of_main_gun_firing	324
	2.1.3.3.10	sound_of_engine_cranking_ start	324
	2.1.3.3.11	sound_of_engine_cranking_ stop	324

	2.1.3.3.12	sound_of_engine_cranking_	
		stall	325
	2.1.3.3.13	sound_of_tracks	325
	2.1.3.3.14	sound_of_engine_start	326
	2.1.3.3.15	sound_of_engine_stop.....	326
	2.1.3.3.16	sound_of_engine	326
	2.1.3.3.17	sound_of_gun_elevating	327
	2.1.3.3.18	sound_of_turret_traversing	327
	2.1.3.3.19	sound_of_turret_power_on	328
	2.1.3.3.20	sound_of_turret_power_	
		already_on	328
	2.1.3.3.21	sound_of_turret_power_off.....	328
	2.1.3.3.22	sound_of_turret_drive_on	329
	2.1.3.3.23	sound_of_turret_drive_	
		already_on	329
	2.1.3.3.24	sound_of_turret_drive_off	329
	2.1.3.3.25	sound_of_engine_accessory_	
		on.....	330
	2.1.3.3.26	sound_of_engine_accessory_	
		already_on	330
	2.1.3.3.27	sound_of_engine_accessory_	
		off	330
	2.1.3.3.28	sound_of_random_sounds.....	331
	2.1.3.3.29	channel_2_check	331
	2.1.3.3.30	channel_5_check	331
2.1.3.4	kato_sound.c.....		332
	2.1.3.4.1	sound_denial_check	333
	2.1.3.4.2	sound_make_veh_spec_	
		sound	333
	2.1.3.4.3	sound_force_veh_spec_sound.....	334
	2.1.3.4.4	sound_init.....	334
	2.1.3.4.5	sound_dont_use.....	334
	2.1.3.4.6	sound_simul	334
	2.1.3.4.7	sound reset.....	335
	2.1.3.4.8	sound_we_just_died	335
	2.1.3.4.9	sound_of_vehicle	335
	2.1.3.4.12	sound_of_random_sounds.....	336
2.1.4	Controls Interface Software		337
2.1.4.1	Controls using IDC Boards		337
	2.1.4.1.1	libidc.....	339
	2.1.4.1.1.1	choose_fifo.c	339
	2.1.4.1.1.1.1	idc_choose_fifo	339
	2.1.4.1.1.2	i_error.c	340
	2.1.4.1.1.2.1	libidc_error_report	340

2.1.4.1.1.3	i_getact.c	341
2.1.4.1.1.3.1	libidc_get_action	341
2.1.4.1.1.4	i_getacts.c	342
2.1.4.1.1.4.1	idc_get_actions	342
2.1.4.1.1.5	i_getdevice.c	342
2.1.4.1.1.5.1	idc_get_device_type	342
2.1.4.1.1.6	i_getnames.c	343
2.1.4.1.1.6.1	idc_get_names	343
2.1.4.1.1.7	i_getoffset.c	344
2.1.4.1.1.7.1	idc_get_offset	344
2.1.4.1.1.8	i_getport.c	344
2.1.4.1.1.8.1	idc_get_port_name	344
2.1.4.1.1.9	i_getstat.c	345
2.1.4.1.1.9.1	idc_get_station_description	345
2.1.4.1.1.10	i_loc.c	345
2.1.4.1.1.11	i_open_port.c	346
2.1.4.1.1.11.1	idc_open_port	346
2.1.4.1.1.12	i_perror.c	347
2.1.4.1.1.12.1	libidc_perror_report	347
2.1.4.1.1.13	i_port_stk.c	347
2.1.4.1.1.14.1	port_stuck	347
2.1.4.1.1.15	i_raw_16_set.c	348
2.1.4.1.1.15.1	idc_raw_16_set_cmd	348
2.1.4.1.1.16	i_raw_16_st2.c	349
2.1.4.1.1.16.1	idc_raw_16_set2_cmd	349
2.1.4.1.1.17	i_raw_set.c	350
2.1.4.1.1.17.1	idc_raw_set_cmd	350
2.1.4.1.1.18	i_readbody.c	351
2.1.4.1.1.18.1	libidc_read_idc_parameter_ body	351
2.1.4.1.1.19	i_readfile.c	352
2.1.4.1.1.19.1	read_idc_parameter_file	352
2.1.4.1.1.20	i_readhead.c	353
2.1.4.1.1.20.1	libidc_read_idc_parameter_ header	353
2.1.4.1.1.21	i_reset.c	354
2.1.4.1.1.21.1	idc_reset_cmd	354
2.1.4.1.1.22	i_strsave.c	355
2.1.4.1.1.22.1	libidc_strsave	355
2.1.4.1.1.23	idc_loc.h	355
2.1.4.1.1.24	init.c	356
2.1.4.1.1.24.1	idc_init	356
2.1.4.1.1.24.2	idc_fifo_init	357

2.1.4.1.1.24.3	idc_fifo_uninit.....	357
2.1.4.1.1.24.4	idc_reset	357
2.1.4.1.1.24.5	idc_reset_input	358
2.1.4.1.1.24.6	idc_reset_output	358
2.1.4.1.1.25	op_16_set.c.....	359
2.1.4.1.1.25.1	idc_output_16_set	359
2.1.4.1.1.26	op_16_set2.c.....	360
2.1.4.1.1.26.1	idc_output_16_set2	360
2.1.4.1.1.27	op_rest.c	361
2.1.4.1.1.27.1	idc_output_restore.....	361
2.1.4.1.1.28	op_rest_c.c.....	362
2.1.4.1.1.28.1	idc_output_restore_cond	362
2.1.4.1.1.29	op_set.c.....	363
2.1.4.1.1.29.1	idc_output_set_ns_cond	363
2.1.4.1.1.30	op_set_c.c.....	364
2.1.4.1.1.30.1	idc_output_set_cond.....	364
2.1.4.1.1.31	op_set_ns.c	365
2.1.4.1.1.31.1	idc_output_set_ns_cond	365
2.1.4.1.1.32	op_set_ns_c.c	366
2.1.4.1.1.32.1	idc_output_set_ns_cond	366
2.1.4.1.1.33	respond.c	367
2.1.4.1.1.33.1	idc_respond	367
2.1.4.1.2	libpots	368
2.1.4.1.2.1	p_clamp.c	368
2.1.4.1.2.1.1	pots_clamp_pot_between	368
2.1.4.1.2.2	p_lcr.c.....	369
2.1.4.1.2.2.1	pots_scale_lcr	369
2.1.4.1.2.3	p_lr_both.c.....	370
2.1.4.1.2.3.1	pots_scale_lr_both.....	370
2.1.4.1.2.4	p_lr_pos.c	371
2.1.4.1.2.4.1	pots_scale_lr_pos	371
2.1.4.1.2.5	p_three.c	372
2.1.4.1.2.5.1	pots_check_three.....	372
2.1.4.1.2.6	p_two.c	373
2.1.4.1.2.6.1	pots_check_two.....	373
2.1.4.1.2.7	libpots.h.....	373
2.1.4.1.3	m1_idc.c	374
2.1.4.1.3.1	idc_get_num_idcs.....	374
2.1.4.1.3.2	idc_array_init	374
2.1.4.1.3.3	idc_invert_outputs.....	374
2.1.4.1.3.4	idc_veh_spec_init.....	374
2.1.4.1.4	m2_idc.c	375
2.1.4.1.4.1	idc_get_num_idcs.....	375

	2.1.4.1.4.2	idc_array_init	375
	2.1.4.1.4.3	idc_invert_outputs.....	375
	2.1.4.1.3.4	idc_veh_spec_init.....	376
	2.1.4.1.3.5	idc_set_reticle_init_val	376
	2.1.4.1.5	kato_idc.c	377
	2.1.4.1.5.1	idc_get_num_idcs.....	377
	2.1.4.1.5.2	idc_array_init	377
	2.1.4.1.5.3	idc_veh_spec_init.....	377
2.1.4.2	High Performance Analog Interface		378
	2.1.4.2.1	libdtad.....	378
	2.1.4.2.1.1	ain.c	378
	2.1.4.2.1.1.1	ain	378
	2.1.4.2.1.2	attach.c	379
	2.1.4.2.1.2.1	dtad_attach	379
	2.1.4.2.1.3	cur_minus12.c	380
	2.1.4.2.1.3.1	current_minus12.....	380
	2.1.4.2.1.4	cur_plus12.c	380
	2.1.4.2.1.4.1	current_plus12.....	380
	2.1.4.2.1.5	cur_plus5.c	381
	2.1.4.2.1.5.1	current_plus5.....	381
	2.1.4.2.1.6	cur_temp.c.....	382
	2.1.4.2.1.6.1	current_temperature	382
	2.1.4.2.1.7	data.c	382
	2.1.4.2.1.8	detach.c.....	383
	2.1.4.2.1.8.1	dtad_detach.....	383
	2.1.4.2.1.9	dtad_loc.h	383
	2.1.4.2.1.10	init.c.....	384
	2.1.4.2.1.10.1	dtad_signal_handler	384
	2.1.4.2.1.10.2	dtad_init.....	384
	2.1.4.2.1.11	uninit.c.....	385
	2.1.4.2.1.11.1	dtad_uninit.....	385
2.1.5	Status Panel Interface Software.....		386
	2.1.5.1	libbbd.....	386
	2.1.5.1.1	attach.c.....	387
	2.1.5.1.1.1	bbd_attach	387
	2.1.5.1.2	bbd_loc.h	387
	2.1.5.1.3	bit_in.c.....	388
	2.1.5.1.3.1	bbd_bit_in	388
	2.1.5.1.4	bit_out.c.....	389
	2.1.5.1.4.1	bbd_bit_out	389
	2.1.5.1.5	byte_in.c	390
	2.1.5.1.5.1	bbd_byte_in.....	390
	2.1.5.1.6	byte_out.c	390

	2.1.5.1.6.1	bbd_byte_out.....	390
	2.1.5.1.7	control_in.c.....	391
	2.1.5.1.7.1	bbd_control_in	391
	2.1.5.1.8	control_out.c.....	391
	2.1.5.1.8.1	bbd_control_out	391
	2.1.5.1.9	data.c	391
	2.1.5.1.10	detach.c.....	392
	2.1.5.1.10.1	bbd_detach	392
	2.1.5.1.11	init.c.....	392
	2.1.5.1.11.1	bbd_init	392
	2.1.5.1.11.2	bbd_signal_handler	393
	2.1.5.1.12	statistics.c	393
	2.1.5.1.13	status.c.....	394
	2.1.5.1.13	status_out.....	394
	2.1.5.1.16	uninit.c.....	395
	2.1.5.1.16.1	bbd_uninit	395
2.1.5.2	ml_status.c.....		396
	2.1.5.2.1	what_is_voltage12P	397
	2.1.5.2.2	what_is_voltage12N.....	397
	2.1.5.2.3	what_is_voltage5.....	397
	2.1.5.2.4	what_is_temperature	397
	2.1.5.2.5	status_preset	397
	2.1.5.2.6	status_init	398
	2.1.5.2.7	status_simul.....	398
	2.1.5.2.8	status_print_temp_and_ supplies.....	398
	2.1.5.2.9	driver_dead.....	398
	2.1.5.2.10	turret_dead.....	399
	2.1.5.2.11	ammo_dead	399
	2.1.5.2.12	cig_dead	399
	2.1.5.2.13	net_dead	400
	2.1.5.2.14	ser_dead.....	400
	2.1.5.2.15	dtad_dead	400
	2.1.5.2.16	sound_dead.....	401
	2.1.5.2.17	plus12_dead.....	401
	2.1.5.2.18	minus12_dead.....	401
	2.1.5.2.19	plus5_dead.....	402
	2.1.5.2.20	enable_status_printing.....	402
	2.1.5.2.21	disable_status_printing.....	402
	2.1.5.2.22	cig_failed_fsm.....	402
	2.1.5.2.23	monitor_status	403
2.1.5.3	m2_status.c.....		404
	2.1.5.3.1	what_is_voltage12P	405

2.1.5.3.2	what_is_voltage12N.....	405
2.1.5.3.3	what_is_voltage5.....	405
2.1.5.3.4	what_is_temperature	406
2.1.5.3.5	status_preset	406
2.1.5.3.6	status_init	406
2.1.5.3.7	status_simul.....	406
2.1.5.3.8	status_print_temp_and_ supplies.....	407
2.1.5.3.9	driver_dead.....	407
2.1.5.3.10	turret_dead.....	407
2.1.5.3.11	cig_dead	408
2.1.5.3.12	net_dead	408
2.1.5.3.13	ser_dead.....	408
2.1.5.3.14	dtad_dead	409
2.1.5.3.15	sound_dead.....	409
2.1.5.3.16	plus12_dead.....	409
2.1.5.3.17	minus12_dead.....	410
2.1.5.3.18	plus5_dead.....	410
2.1.5.3.19	enable_status_printing.....	410
2.1.5.3.20	disable_status_printing.....	410
2.1.5.3.21	cig_failed_fsm.....	411
2.1.5.3.22	monitor_status	412
2.1.5.4	kato_status.c	413
2.1.5.4.1	what_is_voltage12P	414
2.1.5.4.2	what_is_voltage12N.....	414
2.1.5.4.3	what_is_voltage5.....	414
2.1.5.4.4	what_is_temperature	415
2.1.5.4.5	status_preset	415
2.1.5.4.6	status_init	415
2.1.5.4.7	status_simul.....	415
2.1.5.4.8	status_print_temp_and_ supplies.....	415
2.1.5.4.9	hard_dead	416
2.1.5.4.10	soft_dead	416
2.1.5.4.11	cig_dead	416
2.1.5.4.12	net_dead	417
2.1.5.4.13	ser_dead.....	417
2.1.5.4.14	dtad_dead	417
2.1.5.4.15	sound_dead.....	418
2.1.5.4.16	plus12_dead.....	418
2.1.5.4.17	minus12_dead.....	418
2.1.5.4.18	plus5_dead.....	419
2.1.5.4.19	enable_status_printing.....	419

	2.1.5.4.20	disable_status_printing.....	419
	2.1.5.4.21	cig_failed_fsm.....	419
	2.1.5.4.22	monitor_status.....	420
2.1.6	Keyboard Interface Software		421
2.1.6.1	libkeybrd.....		421
	2.1.6.1.1	close.c.....	421
	2.1.6.1.1.1	keybrd_tty_close	422
	2.1.6.1.2	init.c.....	422
	2.1.6.1.2.1	keybrd_tty_init	422
	2.1.6.1.3	key_loc.h	423
	2.1.6.1.4	read.c	424
	2.1.6.1.4.1	keybrd_tty_read.....	424
	2.1.6.1.5	reset.c.....	425
	2.1.6.1.5.1	keybrd_tty_reset.....	425
	2.1.6.1.6	write.c.....	426
	2.1.6.1.6.1	keybrd_tty_write	426
2.1.6.2	m1_keybrd.c.....		427
	2.1.6.2.1	keyboard_really_use.....	428
	2.1.6.2.2	keyboard_use_cupola	428
	2.1.6.2.3	keyboard_init.....	428
	2.1.6.2.4	keyboard_simul	429
	2.1.6.2.5	keyboard_setup_terminal	430
	2.1.6.2.6	keyboard_reset_terminal	430
	2.1.6.2.7	keyboard_exit_gracefully.....	431
2.1.6.3	m2_keybrd.c.....		432
	2.1.6.3.1	keyboard_really_use.....	433
	2.1.6.3.2	keyboard_use_cupola	433
	2.1.6.3.3	keyboard_init.....	433
	2.1.6.3.4	keyboard_simul	434
	2.1.6.3.5	keyboard_setup_terminal	435
	2.1.6.3.6	keyboard_reset_terminal	435
	2.1.6.3.7	keyboard_exit_gracefully.....	436
2.1.6.4	kato_keybrd.c		437
	2.1.6.4.1	keyboard_really_use.....	438
	2.1.6.4.2	keyboard_init.....	438
	2.1.6.4.3	keyboard_simul	439
	2.1.6.4.4	keyboard_setup_terminal	440
	2.1.6.4.5	keyboard_reset_terminal	440
	2.1.6.4.6	keyboard_exit_gracefully.....	440
2.1.7	Spaceball Interface Software.....		441
2.1.7.1	libspaceball.....		441
	2.1.7.1.1	sbcustom.c	441
	2.1.7.1.2	sbllibry.c	442

	2.1.7.1.3	sbtest.c	442
	2.1.7.1.4	sbtute.c	442
	2.1.7.1.5	sbcustom.h	442
	2.1.7.1.6	sbllibry.h	442
2.1.7.2	kato_sb.c		443
	2.1.7.2.1	spaceball_simul	443
	2.1.7.2.2	spaceball_exit	444
	2.1.7.2.3	initialize_spaceball	444
	2.1.7.2.4	display_data	444
	2.1.7.2.5	mypressed	445
	2.1.7.2.6	mytranslate	445
	2.1.7.2.7	myrotate	446
2.2	M1 VEHICLE SIMULATION FUNCTIONS		447
2.2.1	Top Level M1 Simulation Software		448
2.2.1.1	m1_main.c		448
	2.2.1.1.1	print_help	449
	2.2.1.1.2	print_veh_logo	449
	2.2.1.1.3	veh_spec_startup	450
	2.2.1.1.4	veh_spec_idle	450
	2.2.1.1.5	veh_spec_init	451
	2.2.1.1.6	veh_spec_simulate	452
	2.2.1.1.7	veh_spec_stop	452
	2.2.1.1.8	veh_spec_exit	453
	2.2.1.1.9	main	454
	2.2.1.1.10	reconstitute_vehicle	456
2.2.2	M1 Controls/Switchology		457
2.2.2.1	Low Level Control Handling		458
	2.3.2.1.1	m1_ctl_npc.c	459
	2.3.2.1.2	m1_ctl_mpc.c	463
	2.3.2.1.2	m1_ctl_tpc.c	467
2.2.2.2	Finite State Machines		470
	2.2.2.2.1	m1_ctl_fsm.c	470
	2.2.2.2.2	m1_handles.c	472
	2.2.2.2.2.1	handles_simul	473
	2.2.2.2.2.2	handles_gunner_control_ fixed	473
	2.2.2.2.2.3	handles_gunner_control_ broken	473
	2.2.2.2.2.4	handles_commander_control_ fixed	473
	2.2.2.2.2.5	handles_commander_control_ broken	473
	2.2.2.2.2.6	handles_gunner_palm_on	473

2.2.2.2.2.7	handles_gunner_palm_off.....	473
2.2.2.2.2.8	handles_commander_palm_	
	on.....	473
2.2.2.2.2.9	handles_commander_	
	palm_off.....	474
2.2.2.2.2.10	handles_set_gunner_elevation	474
2.2.2.2.2.11	handles_set_commander_	
	elevation.....	474
2.2.2.2.2.12	handles_set_gunner_traverse	474
2.2.2.2.2.13	handles_set_commander_	
	traverse.....	474
2.2.2.2.2.14	handles_gunner_laser_button_	
	released.....	475
2.2.2.2.2.15	handles_gunner_laser_button_	
	depressed.....	475
2.2.2.2.2.16	handles_commander_laser_	
	button_released.....	475
2.2.2.2.2.17	handles_commander_laser_	
	button_depressed.....	475
2.2.2.2.2.18	handles_gunner_trigger_	
	depressed.....	475
2.2.2.2.2.19	handles_commander_trigger_	
	depressed.....	475
2.2.2.2.2.20	handles_gunner_trigger_	
	released.....	475
2.2.2.2.2.21	handles_commander_trigger_	
	released.....	475
2.2.2.2.2.22	handles_init.....	476
2.2.2.2.3	m1_firectl.c.....	477
2.2.2.2.3.1	firectl_init.....	477
2.2.2.2.3.2	firectl_malfunction.....	477
2.2.2.2.3.3	firectl_laser_malfunction_set.....	478
2.2.2.2.3.4	firectl_laser_malfunction_	
	reset.....	478
2.2.2.2.3.5	firectl_ready_to_fire.....	478
2.2.2.2.3.6	firectl_gun_select_main.....	479
2.2.2.2.3.7	firectl_gun_select_safe.....	479
2.2.2.2.3.8	firectl_gun_select_coax.....	479
2.2.2.2.3.9	firectl_gun_select_status.....	479
2.2.2.2.3.10	firectl_fire_control_normal.....	479
2.2.2.2.3.11	firectl_fire_control_	
	emergency.....	479
2.2.2.2.3.12	firectl_fire_control_manual.....	479
2.2.2.2.3.13	firectl_fire_control_status.....	479
2.2.2.2.3.14	firectl_gun_drive_manual.....	480

	2.2.2.2.3.15	firectl_gun_drive_powered	480
	2.2.2.2.3.16	firectl_gun_drive_uncpl	480
	2.2.2.2.3.17	firectl_gun_drive_status	480
2.2.2.3	Specialized Output Devices		481
	2.2.2.3.1	m1_meter.c	481
	2.2.2.3.1.1	meter_init	481
	2.2.2.3.1.2	meter_speed_set	482
	2.2.2.3.1.3	meter_tach_set	482
	2.2.2.3.1.4	meter_fuel_set	483
	2.2.2.3.1.5	meter_volt_set	483
	2.2.2.3.1.6	meter_select_front_left_tank	483
	2.2.2.3.1.7	meter_select_front_right_tank	484
	2.2.2.3.1.8	meter_select_rear_tank	484
	2.2.2.3.2	m1_pots.c	485
	2.2.2.3.2.1	pots_init	486
	2.2.2.3.2.2	pots_comm_trav_real	487
	2.2.2.3.2.3	pots_comm_elev_real	487
	2.2.2.3.2.4	pots_gunn_trav_real	488
	2.2.2.3.2.5	pots_gunn_elev_real	488
	2.2.2.3.2.6	pots_steer_bar_real	489
	2.2.2.3.2.7	pots_throttle_real	489
	2.2.2.3.2.8	pots_service_brake_real	490
	2.2.2.3.2.9	pots_comm_weap_real	490
	2.2.2.3.2.10	pots_load_peri_real	491
2.2.3	M1 Weapons		492
	2.2.3.1	m1_bcs.c	493
	2.2.3.1.1	bcs_dump_lead_buffer	494
	2.2.3.1.2	bcs_init	494
	2.2.3.1.3	bcs_simul	495
	2.2.3.1.4	bcs_manual_range_ battlesight	495
	2.2.3.1.5	bcs_manual_range_add_ pushed	495
	2.2.3.1.6	bcs_manual_range_drop_ pushed	495
	2.2.3.1.7	bcs_manual_range_released	495
	2.2.3.1.8	bcs_ammo_select_heat	495
	2.2.3.1.9	bcs_ammo_select_apds	496
	2.2.3.1.10	bcs_ammo_select_other	496
	2.2.3.1.11	bcs_range_is	496
	2.2.3.1.12	calc_avg_slew_rate	496
	2.2.3.1.13	bcs_set_ballistics_computer	496
	2.2.3.1.14	bcs_get_lead_azimuth	497

	2.2.3.1.15	bcs_get_super_elevation	497
	2.2.3.1.16	bcs_get_range.....	498
	2.2.3.1.17	bcs_get_time_of_flight.....	498
	2.2.3.1.18	bcs_get_ammo_type_indexed	498
	2.2.3.1.19	bcs_get_range_str.....	498
	2.2.3.1.20	bcs_boot_computer	498
	2.2.3.1.21	bcs_computer_status	499
	2.2.3.1.22	bcs_turn_computer_off	499
	2.2.3.1.23	bcs_check_bootup	500
	2.2.3.2	m1_laser.c	501
	2.2.3.2.1	laser_init.....	502
	2.2.3.2.2	laser_show_status.....	502
	2.2.3.2.3	laser_lrf_failure	502
	2.2.3.2.4	laser_repair_lrf	503
	2.2.3.2.5	laser_power_off.....	503
	2.2.3.2.6	laser_select_safe.....	503
	2.2.3.2.7	laser_select_first_return	504
	2.2.3.2.8	laser_select_last_return	504
	2.2.3.2.9	laser_power_up_safe.....	504
	2.2.3.2.10	laser_power_up_first_return	505
	2.2.3.2.11	laser_power_up_last_return	505
	2.2.3.2.12	time_n_lases_ago	506
	2.2.3.2.13	record_this_lase.....	506
	2.2.3.2.14	laser_perform_lase	507
	2.2.3.2.15	laser_lase.....	507
	2.2.3.2.16	laser_multiple_returns.....	508
	2.2.3.2.17	laser_ready_to_fire.....	508
	2.2.3.2.18	laser_last_return	508
	2.2.3.2.19	laser_range	508
	2.2.3.2.20	laser_fire_control_ malfunction.....	509
	2.2.3.3	m1_weapons.c.....	510
	2.2.3.3.1	weapons_download_ ballistics_tables	511
	2.2.3.3.2	weapons_init.....	511
	2.2.3.3.3	weapons_simul	511
	2.2.3.3.4	weapons_disable_main_gun.....	511
	2.2.3.3.5	weapons_repair_main_gun.....	511
	2.2.3.3.6	weapons_fire_main_gun	513
2.2.4	M1 Failures		514
	2.2.4.1	m1_failure.c.....	516
	2.2.4.1.1	fail_init	517
	2.2.4.1.2	failure_mob_electrical_fixed	517

	2.2.4.1.3	failure_engine_fixed.....	517
	2.2.4.1.4	failure_transmission_fixed.....	518
	2.2.4.1.5	failure_fuel_or_brakes_fixed.....	518
	2.2.4.1.6	failure_fire_control_fixed.....	518
	2.2.4.1.7	failure_gun_turret_fixed.....	518
	2.2.4.1.8	failure_collision_damages.....	519
	2.2.4.1.9	failure_check_cat_kill.....	520
	2.3.4.1.10	failure_check_indir_fire_damages.....	520
2.2.4.2	m1_repair.c.....		521
	2.2.4.2.1	repair_request.....	522
	2.2.4.2.2	repair_simul.....	522
	2.2.4.2.3	repair_init.....	523
	2.2.4.2.4	clear_repair_vehicles.....	523
	2.2.4.2.5	repair_near_repair.....	523
	2.2.4.2.6	send_feed_me_packets_repair_vehicles.....	523
	2.2.4.2.7	repair_quiet_state.....	524
	2.2.4.2.8	repair_request_state.....	524
	2.2.4.2.9	print_repair_status.....	525
2.2.5	M1 Munitions Management.....		526
2.2.5.1	m1_ammo.c.....		527
	2.2.5.1.1	ammo_init.....	529
	2.2.5.1.2	ammo_simul.....	529
	2.2.5.1.3	ammo_check_autoloader_unload.....	529
	2.2.5.1.4	ammo_check_autoloader_load.....	530
	2.2.5.1.5	ammo_init_ammo_racks.....	530
	2.2.5.1.6	ammo_supply_full.....	531
	2.2.5.1.7	ammo_supply_empty.....	531
	2.2.5.1.8	ammo_loaders_arms.....	531
	2.2.5.1.9	ammo_knee_switch_on.....	532
	2.2.5.1.10	ammo_knee_switch_off.....	532
	2.2.5.1.11	ammo_tube_selected.....	533
	2.2.5.1.12	ammo_arm_panel_check.....	533
	2.2.5.1.13	ammo_resupply_check.....	534
	2.2.5.1.14	ammo_get_quantity.....	535
	2.2.5.1.15	ammo_transfer_semi_heat.....	535
	2.2.5.1.16	ammo_transfer_semi_apds.....	536
	2.2.5.1.17	ammo_transfer_hull_heat.....	536
	2.2.5.1.18	ammo_transfer_hull_apds.....	536
	2.2.5.1.19	ammo_transfer_no_transfer.....	537

2.2.5.1.20	ammo_transfer_redist_send	537
2.2.5.1.21	ammo_transfer_redist_recv	537
2.2.5.1.22	ammo_get_transfer_status	538
2.2.5.1.23	ammo_breech_pushed	538
2.2.5.1.24	ammo_breech_unload_ pushed	538
2.2.5.1.25	ammo_type_loaded_quick	538
2.2.5.1.26	ammo_ejection_guard_armed	539
2.2.5.1.27	ammo_ejection_guard_safe	540
2.2.5.1.28	ammo_ejection_guard_status	540
2.2.5.1.29	ammo_open_blast_door	540
2.2.5.1.30	ammo_close_blast_door	541
2.2.5.1.31	ammo_gun_fired	541
2.2.5.1.32	ammo_ready_to_fire	541
2.2.5.1.33	ammo_get_semi_heat_ quantity	542
2.2.5.1.34	ammo_get_semi_apds_ quantity	542
2.2.5.1.35	ammo_get_hull_heat_ quantity	542
2.2.5.1.36	ammo_get_hull_apds_ quantity	542
2.2.5.1.37	ammo_get_ready_heat_ quantity	543
2.2.5.1.38	ammo_get_ready_apds_ quantity	543
2.2.5.1.39	ammo_get_heat105_quantity	543
2.2.5.1.40	ammo_get_apds105_quantity	543
2.2.5.1.41	ammo_add_round	544
2.2.5.1.42	ammo_subtract_round	544
2.2.5.1.43	ammo_blast_door_open	544
2.2.5.1.44	ammo_turret_power_on	544
2.2.5.1.45	ammo_turret_power_off	544
2.2.5.1.46	ammo_breech_ready	545
2.2.5.1.47	ammo_start_loader_timer	545
2.2.5.1.48	ammo_stop_loader timer	545
2.2.5.1.49	ammo_start_blast_door_timer	545
2.2.5.1.50	ammo_stop_blast_door_timer	546
2.2.5.1.51	ammo_blast_door_check	546
2.2.5.1.52	ammo_flash_check	546
2.2.5.1.53	ammo_resupply_receive_ timeout_check	547
2.2.5.1.54	ammo_start_resupply_ receive_timer	547

2.2.5.1.55	ammo_stop_resupply_	
	receive_timer	548
2.2.5.1.56	ammo_change_resupply	548
2.2.5.1.57	ammo_stop_resupply	548
2.2.5.1.58	ammo_start_internal_	
	resupply	549
2.2.5.1.59	ammo_start_external_	
	resupply	549
2.2.5.1.60	ammo_start_external_send	550
2.2.5.1.61	ammo_decide_resupply_	
	receive	550
2.2.5.1.62	ammo_decide_receive_	
	location	551
2.2.5.1.63	ammo_decide_round_type	551
2.2.5.1.64	ammo_stop_timers	551
2.2.5.1.65	ammo_restore_ammo	552
2.2.5.1.66	ammo_resupply_sent	552
2.2.5.1.67	ammo_decide_resupply_send	553
2.2.5.1.68	ammo_decide_resupply_slot	553
2.2.5.1.69	ammo_print_statistics	553
2.2.5.1.70	ammo_enable_autoloader	553
2.2.5.2	m1_fuelsys.c	554
2.2.5.2.1	fuel_init_tanks	555
2.3.5.2.2	fuel_init	555
2.2.5.2.3	fuel_simul	555
2.2.5.2.4	fuel_meter_value	556
2.2.5.2.5	fuel_warning_levels	556
2.2.5.2.6	fuel_check_xfer_timer	556
2.2.5.2.7	fuel_rear_tank_not_empty	557
2.2.5.2.8	fuel_set_flow	557
2.2.5.2.9	fuel_select_front_left_tank	557
2.2.5.2.10	fuel_select_front_right_tank	558
2.2.5.2.11	fuel_select_rear_tank	558
2.2.5.2.12	fuel_xfer_fuel	558
2.2.5.2.13	fuel_master_power_on	558
2.2.5.2.14	fuel_master_power_off	559
2.2.5.2.15	fuel_level_rear	559
2.2.5.2.16	fuel_level_left	559
2.2.5.2.17	fuel_level_right	559
2.2.5.2.18	fuel_repair_transfer_pump	560
2.2.5.2.19	fuel_transfer_pump_failure	560
2.2.5.2.20	fuel_supply_full	560
2.2.5.2.21	fuel_decide_resupply_	
	quantity	561

	2.2.5.2.22	fuel_start_external_resupply	561
	2.2.5.2.23	fuel_stop_resupply	561
	2.2.5.2.24	fuel_resupply_tank	562
2.2.5.3	m1_resupp.c		563
	2.2.5.3.1	clear_ammo_carriers	564
	2.2.5.3.2	clear_fuel_carriers	564
	2.2.5.3.3	clear_ammo_receivers	564
	2.2.5.3.4	print_resupply_status	564
	2.2.5.3.5	send_feed_me_packets_ ammo_carriers	565
	2.2.5.3.6	send_feed_me_packets_ fuel_carriers	565
	2.2.5.3.7	resupply_near_ammo_carrier	565
	2.2.5.3.8	resupply_near_fuel_carrier	566
	2.2.5.3.9	resupply_near_ammo_ receiver	566
	2.2.5.3.10	resupply_ammo_received	566
	2.2.5.3.11	resupply_fuel_received	566
	2.2.5.3.12	resupply_offer_packet	566
	2.2.5.3.13	resupply_thank_you_packet	568
	2.2.5.3.14	resupply_feed_me_packet	569
	2.2.5.3.15	resupply_gating_conditions	570
	2.2.5.3.16	ammo_receive_quiet_state	570
	2.2.5.3.17	fuel_receive_quiet_state	571
	2.2.5.3.18	ammo_send_quiet_state	571
	2.2.5.3.19	ammo_receive_request_state	572
	2.2.5.3.20	fuel_receive_request_state	573
	2.2.5.3.21	ammo_send_waiting_state	574
	2.2.5.3.22	ammo_receive_loading_state	575
	2.2.5.3.23	fuel_receive_loading_state	576
	2.2.5.3.24	ammo_send_servicing_state	577
	2.2.5.3.25	ammo_resupply_receive_ simul	577
	2.2.5.3.26	fuel_resupply_receive_simul	578
	2.2.5.3.27	ammo_resupply_send_simul	578
	2.2.5.3.28	resupply_init	578
	2.2.5.3.29	resupply_simul	579
	2.2.5.3.29	service_check_vehicle_type	579
	2.2.5.3.30	resupply_stop_ammo_ resupply	580
	2.2.5.3.31	resupply_stop_fuel_resupply	580
	2.2.5.3.32	resupply_offer_canceled	580
	2.2.5.3.33	resupply_request_canceled	580
	2.2.5.3.34	vehicle_is_close	581

2.2.6	M1 Vehicle Model	582
2.2.6.1	Internal Kinematics	582
2.2.6.1.1	m1_turret.c	583
2.2.6.1.1.1	turret_init.....	585
2.2.6.1.1.2	turret_simul	586
2.2.6.1.1.3	turret_move	586
2.2.6.1.1.4	turret_get_gps_slew_rate	587
2.2.6.1.1.5	turret_get_turret_slew_rate	587
2.2.6.1.1.6	turret_handles_values.....	587
2.2.6.1.1.7	turret_calc_turret_slew.....	588
2.2.6.1.1.9	calc_slew_from_handles	589
2.2.6.1.1.10	turret_calc_gun_elev	590
2.2.6.1.1.11	calc_elev_from_handle	591
2.2.6.1.1.12	turret_gyros_simul	591
2.2.6.1.1.13	turret_gyros_spool_up.....	592
2.2.6.1.1.14	turret_gyros_spool_down.....	592
2.2.6.1.1.15	turret_gyros_status	592
2.2.6.1.1.16	turret_break_gearbox	592
2.2.6.1.1.17	turret_repair_gearbox.....	592
2.2.6.1.1.18	turret_break_elevation_drive	592
2.2.6.1.1.19	turret_repair_elevation_drive.....	592
2.2.6.1.1.20	turret_break_stab_system.....	592
2.2.6.1.1.21	turret_repair_stab_system	592
2.2.6.1.1.22	turret_break_mount_interface	593
2.2.6.1.1.23	turret_repair_mount_interface.....	593
2.2.6.1.1.24	turret_break_traverse_drive	593
2.2.6.1.1.25	turret_repair_traverse_drive.....	593
2.2.6.1.1.26	turret_fire_control_	
	emergency	593
2.2.6.1.1.27	turret_fire_manual.....	593
2.2.6.1.1.28	turret_fire_control_normal.....	593
2.2.6.1.1.29	turret_gun_turret_drive_	
	uncoupled	593
2.2.6.1.1.30	turret_gun_turret_drive_	
	powered	594
2.2.6.1.1.31	turret_gun_turret_drive_	
	manual	594
2.2.6.1.1.32	turret_collision_detected	594
2.2.6.1.1.33	make_sound_of_no_slewing.....	595
2.2.6.1.1.34	make_sound_of_no_elevating.....	595
2.2.6.1.1.35	make_sound_of_no_	
	turret_noise.....	595
2.2.6.1.1.36	turret_get_gun_to_world.....	596
2.2.6.1.2	m1_cupola.c	597

	2.2.6.1.2.1	convert_disp_to_angle	597
	2.2.6.1.2.2	cupola_cws_new_value.....	598
	2.2.6.1.2.3	cupola_lscope_new_value.....	598
	2.2.6.1.2.4	cupola_simul	598
	2.2.6.1.2.5	cupola_init.....	598
2.2.6.2	Propulsion Simulation		599
	2.2.6.2.1	m1_dtrain.c.....	599
	2.2.6.2.1.1	drivetrain_load_torque_ converter.....	601
	2.2.6.2.1.2	drivetrain_lockup_clutch.....	602
	2.2.6.2.1.3	drivetrain_torque_ converter_speed.....	602
	2.2.6.2.1.4	drivetrain_neutral	602
	2.2.6.2.1.5	drivetrain_low	602
	2.2.6.2.1.6	drivetrain_drive	602
	2.2.6.2.1.7	drivetrain_reverse.....	602
	2.2.6.2.1.8	drivetrain_pivot	602
	2.2.6.2.1.9	drivetrain_set_steering_bar	603
	2.2.6.2.1.10	drivetrain_set_service_brake.....	603
	2.2.6.2.1.11	drivetrain_set_parking_brake.....	603
	2.2.6.2.1.12	drivetrain_release_parking_ brake	603
	2.2.6.2.1.13	drivetrain_service_brake_ failure	603
	2.2.6.2.1.14	drivetrain_parking_brake_ failure	604
	2.2.6.2.1.15	drivetrain_repair_service_ brake	604
	2.2.6.2.1.16	drivetrain_repair_parking_ brake	604
	2.2.6.2.1.17	drivetrain_transmission_ select_neutral.....	604
	2.2.6.2.1.18	load_sprocket	604
	2.2.6.2.1.19	compute_fd_brake_torque.....	605
	2.2.6.2.1.20	get_braking_factor	605
	2.2.6.2.1.21	load_final_drive	605
	2.2.6.2.1.22	set_gear_ratio	605
	2.2.6.2.1.23	gearbox_shift.....	606
	2.2.6.2.1.24	load_gearbox	606
	2.2.6.2.1.25	power_gearbox	606
	2.2.6.2.1.26	current_fd_inertia.....	606
	2.2.6.2.1.27	power_final_drive	607
	2.2.6.2.1.28	differential steer.....	607
	2.2.6.2.1.29	power_sprocket	608

2.2.6.2.1.30	power_engine	608
2.2.6.2.1.31	compute_lumped_inertias	608
2.2.6.2.1.32	compute_compile_time_ constants.....	608
2.2.6.2.1.33	power_torque_converter.....	609
2.2.6.2.1.34	load_drivetrain	609
2.2.6.2.1.35	power_drivetrain	609
2.2.6.2.1.36	send_transmission_oil_status.....	610
2.2.6.2.1.37	send_trans_maintenance_ status.....	610
2.2.6.2.1.38	send_dtrain_outputs	610
2.2.6.2.1.39	transmission_oil_system_ simul.....	611
2.2.6.2.1.40	drivetrain_simul	611
2.2.6.2.1.41	drivetrain_init.....	611
2.2.6.2.1.42	drivetrain_clog_transmission_ oil_filter.....	611
2.2.6.2.1.43	drivetrain_replace_ transmission_oil_filter.....	612
2.2.6.2.1.44	drivetrain_transmission_ oil_leak	612
2.2.6.2.1.45	drivetrain_repair_ transmission_oil_leak.....	612
2.2.6.2.1.46	drivetrain_refill_ transmission_oil	612
2.2.6.2.1.47	drivetrain_replace_ transmission.....	612
2.2.6.2.1.48	drivetrain_transmission_ failure	612
2.2.6.2.2	ml_engine.c	613
2.2.6.2.2.1	set_power	614
2.2.6.2.2.2	compute_fuel_consumption	615
2.2.6.2.2.3	engine_dynamics.....	615
2.2.6.2.2.4	send_engine_sound	615
2.2.6.2.2.5	send_engine_controls_status.....	616
2.2.6.2.2.6	send_all_outputs.....	616
2.2.6.2.2.7	engine_oil_system_simul	617
2.2.6.2.2.7	engine_simul	617
2.2.6.2.2.8	engine_running.....	617
2.2.6.2.2.9	engine_spooling_up	617
2.2.6.2.2.10	engine_spooling_down.....	618
2.2.6.2.2.11	engine_get_speed	618
2.2.6.2.2.12	engine_get_torque	618
2.2.6.2.2.13	engine_get_power	618
2.2.6.2.2.14	engine_get_max_power	619

2.2.6.2.2.15	engine_tac_idle_switch_on	619
2.2.6.2.2.16	engine_tac_idle_switch_on	619
2.2.6.2.2.17	engine_set_throttle	619
2.2.6.2.2.18	engine_start_switch	619
2.2.6.2.2.19	spool_down_engine	620
2.2.6.2.2.20	engine_shutoff_switch	620
2.2.6.2.2.21	compute_engine_compile_ time_constants	620
2.2.6.2.2.22	engine_init	620
2.2.6.2.2.23	engine_major_failure	620
2.2.6.2.2.24	engine_replace_powerpack	621
2.2.6.2.2.25	engine_runaway_condition	621
2.2.6.2.2.26	engine_fix_runaway_ condition	621
2.2.6.2.2.27	starter_failure	621
2.2.6.2.2.28	engine_replace_starter	621
2.2.6.2.2.29	engine_pilot_relay_failure	621
2.2.6.2.2.30	engine_replace_pilot_relay	621
2.2.6.2.2.31	engine_clog_oil_filter	621
2.2.6.2.2.32	engine_replace_oil_filter	621
2.2.6.2.2.33	engine_oil_leak	622
2.2.6.2.2.34	engine_degrade_engine_ power	622
2.2.6.2.2.35	engine_refill_oil	622
2.2.6.2.2.36	engine_cooling_system_ failure	622
2.2.6.2.2.37	engine_repair_cooling_ system	622
2.2.6.2.2.38	engine_clog_fuel_filter	622
2.2.6.2.2.39	engine_replace_fuel_filter	622
2.2.6.2.3	ml_tracks.c	623
2.2.6.2.3.1	tracks_compute_friction_ factor	625
2.2.6.2.3.2	tracks_compute_slipping_ state	625
2.2.6.2.3.3	tracks_compute_gravity_load	626
2.2.6.2.3.4	tracks_compute_drag_load	626
2.2.6.2.3.5	tracks_repair_thrown_tracks	627
2.2.6.2.3.6	tracks_throw_left_track	627
2.2.6.2.3.7	tracks_throw_right_track	627
2.2.6.2.3.8	tracks_compute_weight	627
2.2.6.2.3.9	tracks_compute_real_velocity	627
2.2.6.2.3.10	tracks_compute_velocity	628
2.2.6.2.3.11	odometer_simul	628

2.2.6.2.3.12	tracks_set_initial_distance_	
	km.....	628
2.2.6.2.3.13	vehicle_get_elapsed_km	628
2.2.6.2.3.14	tracks_send_velocities.....	629
2.2.6.2.3.15	tracks_stop_drivetrain	629
2.2.6.2.3.16	tracks_init.....	629
2.2.6.2.3.17	tracks_compute_vehicle_	
	force.....	629
2.2.6.2.3.18	compute_actual_vehicle_	
	motion	630
2.2.6.2.3.19	tell_kinematics	630
2.2.6.2.3.20	get_current_soil_type	630
2.2.6.2.3.21	check_for_thrown_track.....	631
2.2.6.2.3.22	send_track_sound	631
2.2.6.2.3.23	send_tracks_outputs	631
2.2.6.2.3.24	get_dust_cloud	632
2.2.6.2.3.25	tracks_simul	632
2.2.6.2.3.26	tracks_motion_disabled.....	632
2.2.6.2.3.27	check_turning_sounds.....	633
2.2.6.2.3.28	tracks_return_slip_state.....	633
2.2.6.3	Vehicle Subsystems.....	634
2.2.6.4.1	m1_electsys.c	634
2.2.6.3.1.1	electsys_simul	637
2.2.6.3.1.2	electsys_dead.....	638
2.2.6.3.1.3	electsys_charge_battery	638
2.2.6.3.1.4	electsys_power_request.....	638
2.2.6.3.1.5	electsys_power_off.....	638
2.2.6.3.1.6	electsys_rpms_to_volts	639
2.2.6.3.1.7	electsys_discharge_battery.....	639
2.2.6.3.1.8	electsys_engine_start_request	640
2.2.6.3.1.9	electsys_aux_pump_request.....	640
2.2.6.3.1.10	electsys_laser_start_request	641
2.2.6.3.1.11	electsys_get_battery_voltage	641
2.2.6.3.1.12	electsys_replace_alternator	641
2.2.6.3.1.13	electsys_alternator_failure	641
2.2.6.3.1.14	electsys_recharge_battery	642
2.2.6.3.1.15	electsys_replace_battery.....	642
2.2.6.3.1.16	electsys_handle_leaky_	
	battery.....	642
2.2.6.3.1.17	electsys_battery_failure.....	642
2.2.6.3.1.18	electsys_vars_status	642
2.2.6.3.1.19	electsys_init.....	643
2.2.6.4.2	m1_hydrsys.c.....	644
2.2.6.4.2.1	hydraulic_simul.....	646

2.2.6.4.2.2	hydraulic_init	647
2.2.6.4.2.3	hydraulic_check_acc	647
2.2.6.4.2.4	hydraulic_deplete_reservoir	647
2.2.6.4.2.5	hydraulic_main_pump_fill	647
2.2.6.4.2.6	hydraulic_aux_pump_fill	648
2.2.6.4.2.7	hydraulic_ammo_door_open_ request	648
2.2.6.4.2.8	hydraulic_ammo_door_ closed.....	648
2.2.6.4.2.9	hydraulic_parking_brake_on_ request	649
2.2.6.4.2.10	hydraulic_slew_turret_ request	649
2.2.6.4.2.11	hydraulic_elevate_gun_ request	650
2.2.6.4.2.12	hydraulic_fraction_flow_rate	650
2.2.6.4.2.13	hydraulic_delta_pressure_ calc	651
2.2.6.4.2.14	hydraulic_master_power_on	651
2.2.6.4.2.15	hydraulic_master_power_off	651
2.2.6.4.2.16	hydraulic_repair_main_pump	651
2.2.6.4.2.17	hydraulic_repair_aux_pump.....	651
2.2.6.4.2.18	hydraulic_main_pump_ failure	651
2.2.6.4.2.19	hydraulic_aux_pump_failure	652
2.2.6.4.2.20	hydraulic_charge_reborn.....	652
2.2.6.4.2.21	hydrsys_vars_status.....	652
2.2.6.4.3	m1_vision.c	653
2.2.6.4.3.1	set_gunners_state	654
2.2.6.4.3.2	set_vision_state	654
2.2.6.4.3.3	set_drivers_state	654
2.2.6.4.3.4	vision_cmdrs_pitch	654
2.2.6.4.3.5	vision_loaders_pitch.....	655
2.2.6.4.3.6	vision_cmdrs_binoculars.....	655
2.2.6.4.3.7	vision_restore_all_blocks.....	655
2.2.6.4.3.8	vision_break_all_blocks.....	656
2.2.6.4.3.9	vision_break_gps.....	656
2.2.6.4.3.10	vision_break_driver_blocks	656
2.2.6.4.3.11	vision_break_driver_center_ block	656
2.2.6.4.3.12	vision_break_cmdrs_blocks	657
2.2.6.4.3.13	vision_break_ldrs_pscope	657
2.2.6.4.3.14	vision_restore_gps.....	657
2.2.6.4.3.15	vision_restore_driver_blocks	657
2.2.6.4.3.16	vision_restore_cmdrs_blocks	658

2.2.6.4.3.17	vision_restore_ldrs_pscope	658
2.2.6.4.3.18	vision_get_dvr_rt_vp	658
2.2.6.4.3.19	vision_get_dvr_ctr_vp.....	658
2.2.6.4.3.20	vision_get_dvr_lt_vp.....	659
2.2.6.4.3.21	vision_get_gnr_vp.....	659
2.2.6.4.3.22	vision_set_otw_night_vision.....	659
2.2.6.4.3.23	vision_set_gunner_white_hot_thermal.....	659
2.2.6.4.3.24	vision_set_driver_white_hot_thermal.....	659
2.2.6.4.3.25	vision_set_gunner_black_hot_thermal.....	660
2.2.6.4.3.26	vision_set_driver_black_hot_thermal.....	660
2.2.6.4.3.27	get_catc_mode.....	660
2.2.6.4.3.28	set_catc_mode	660
2.2.6.4.3.29	get_vision_state.....	660
2.2.6.4.3.30	vision_set_gunner_no_thermal.....	661
2.2.6.4.3.31	vision_set_driver_no_thermal.....	661
2.2.6.4.3.32	toggle_driver_vision_state	661
2.2.6.4.3.33	toggle_gunner_vision_state.....	662
2.2.6.4.3.34	print_view_modes.....	662
2.2.6.4.3.35	vision_init.....	662
2.2.6.4.4	m1_thermal.c.....	663
2.2.6.4.4.1	thermal_init	664
2.2.6.4.4.2	thermal_simul.....	664
2.2.6.4.4.3	thermal_mode_on.....	664
2.2.6.4.4.4	thermal_mode_standby	665
2.2.6.4.4.5	thermal_mode_off	665
2.2.6.4.4.6	thermal_white_hot.....	665
2.2.6.4.4.7	thermal_black_hot.....	666
2.2.6.4.4.8	thermal_3x.....	666
2.2.6.4.4.9	thermal_10x.....	666
2.2.6.4.4.10	thermal_view_on.....	666
2.2.6.4.4.11	thermal_shutter.....	667
2.2.6.4.4.12	thermal_clear	667
2.2.6.4.4.13	start_timing_cooldown_delay	667
2.2.6.4.4.14	start_timing_warmup_delay.....	667
2.2.6.4.4.15	turn_on_gunners_thermal_view	668
2.2.6.4.4.16	turn_off_gunners_thermal_view	668
2.2.6.4.4.17	stop_cooldown_timer	668

	2.2.6.4.4.18	stop_heatup_timer	669
	2.2.6.4.4.19	thermal_cooldown_timeout_	
		check.....	669
	2.2.6.4.4.20	thermal_warmup_timeout_	
		check.....	669
2.2.7	Network Interactions		670
	2.2.7.1	m1_network.c	671
	2.2.7.1.1	send_equipment_status.....	672
	2.2.7.1.2	fill_vehicle_spec_status	673
	2.2.7.1.3	fill_vehicle_spec_appearance.....	673
	2.2.7.1.4	network_process_activation_	
		parameters	674
	2.2.7.1.5	app_init.....	674
	2.2.7.1.6	veh_spec_activate_time	674
2.3	M2 VEHICLE SIMULATION FUNCTIONS		675
2.3.1	M2 Top Level Software		676
	2.3.1.2	m2_main.c	676
	2.3.1.1.1	silent_mode_on	677
	2.3.1.1.2	silent_mode_off.....	677
	2.3.1.1.3	print_help	677
	2.3.1.1.4	print_veh_logo	677
	2.3.1.1.5	veh_spec_startup	678
	2.3.1.1.6	veh_spec_idle	678
	2.3.1.1.7	veh_spec_init.....	679
	2.3.1.1.8	veh_spec_simulate	680
	2.3.1.1.9	veh_spec_stop	680
	2.3.1.1.10	veh_spec_exit.....	681
	2.3.1.1.11	main.....	682
	2.3.1.1.12	reconstitute_vehicle.....	683
2.3.2	M2 Controls/Switchology		684
	2.3.2.1	Low Level Control Handling	685
	2.3.2.1.1	m2_ctl_tdc.c	685
	2.3.2.1.2	m2_ctl_mpc.c	688
	2.3.2.1.3	m2_ctl_tpc.c	691
	2.3.2.1.4	m2_ctl_hnp.c	694
	2.3.2.1.5	m2_ctl_tnp.c.....	696
	2.3.2.2	Finite State Machines	699
	2.3.2.2.1	m2_ctl_fsm.c	699
	2.3.2.2.2	m2_handles.c.....	702
	2.3.2.2.2.1	handles_simul.....	703
	2.3.2.2.2.2	handles_gunner_control_	
		fixed.....	703
	2.3.2.2.2.3	handles_gunner_control_	
		broken.....	703

2.3.2.2.2.4	handles_commander_control_	
	fixed.....	703
2.3.2.2.2.5	handles_commander_control_	
	broken.....	703
2.3.2.2.2.6	handles_gunner_palm_on.....	703
2.3.2.2.2.7	handles_gunner_palm_off.....	703
2.3.2.2.2.8	handles_commander_palm_	
	on.....	704
2.3.2.2.2.9	handles_commander_	
	palm_off.....	704
2.3.2.2.2.10	handles_gunner_fast_slew_on	704
2.3.2.2.2.11	handles_gunner_fast_	
	slew_off.....	704
2.3.2.2.2.12	handles_commander_fast_	
	slew_on.....	704
2.3.2.2.2.13	handles_commander_fast_	
	slew_off.....	704
2.3.2.2.2.14	handles_set_gunner_elevation	704
2.3.2.2.2.15	handles_set_commander_	
	elevation	705
2.3.2.2.2.16	handles_set_gunner_traverse	705
2.3.2.2.2.17	handles_set_commander_	
	traverse	705
2.3.2.2.2.18	handles_gunner_trigger_	
	depressed	705
2.3.2.2.2.19	handles_commander_trigger_	
	depressed	705
2.3.2.2.2.20	handles_gunner_trigger_	
	released.....	705
2.3.2.2.2.21	handles_commander_trigger_	
	released.....	706
2.3.2.2.2.22	handles_launcher_up.....	706
2.3.2.2.2.23	handles_launcher_down.....	706
2.3.2.2.2.24	handles_launcher_idle.....	706
2.3.2.2.2.25	handles_launcher_check.....	706
2.3.2.2.2.26	handles_launcher_default.....	706
2.3.2.2.2.27	handles_init	707
2.3.2.2.3	m2_firectl.c	708
2.3.2.2.3.1	firectl_init.....	708
2.3.2.2.3.2	firectl_simul	708
2.3.2.2.3.3	firectl_gps_mag_4x.....	708
2.3.2.2.3.4	firectl_gps_mag_12x.....	708
2.3.2.2.3.5	firectl_gps_mag_status.....	709
2.3.2.2.3.6	firectl_arm	709
2.3.2.2.3.7	firectl_safe.....	709
2.3.2.2.3.8	firectl_arm_safe_reset_status	709

	2.3.2.2.3.9	firectl_weapon_removed.....	709
	2.3.2.2.3.10	firectl_weapon_ready.....	710
	2.3.2.2.3.11	firectl_25mm_ready_to_fire.....	710
	2.3.2.2.3.12	firectl_tow_ready_to_fire.....	711
2.3.2.3	Specialized Output Devices		712
	2.3.2.3.1	m2_alpha.c	712
	2.3.2.3.1.1	alpha_init.....	713
	2.3.2.3.1.2	alpha_reset.....	713
	2.3.2.3.1.3	alpha_send_mils.....	713
	2.3.2.3.1.4	alpha_send_load.....	714
	2.3.2.3.1.5	alpha_send.....	714
	2.3.2.3.1.6	alpha_get_load	714
	2.3.2.3.2	m2_gages.c.....	715
	2.3.2.3.2.1	gage_oil_pressure.....	716
	2.3.2.3.2.2	gage_oil_temperature.....	716
	2.3.2.3.2.3	gage_coolant_temperature	717
	2.3.2.3.3	m2_meter.c.....	718
	2.3.2.3.3.1	meter_init	718
	2.3.2.3.3.2	meter_speed_set	719
	2.3.2.3.3.3	meter_fuel_set.....	719
	2.3.2.3.3.4	meter_volt_set.....	720
	2.3.2.3.3.5	meter_temp_set	720
	2.3.2.3.3.6	meter_press_set.....	721
	2.3.2.3.4	m2_odom.c.....	722
	2.3.2.3.4.1	odometer_init	723
	2.3.2.3.4.2	odometer_simul.....	723
	2.3.2.3.4.3	odom_set_initial_distance_ km.....	723
	2.3.2.3.4.4	vehicle_get_elapsed_km	723
	2.3.2.3.4.5	odometer_get_elapsed_km.....	724
	2.3.2.3.4.6	odometer_get_elapsed_miles	724
	2.3.2.3.4.7	odometer_mile_counter_reset	724
	2.3.2.3.3.8	odometer_mile_counter.....	724
	2.3.2.3.5	m2_pots.c	725
	2.3.2.3.5.1	pots_init.....	726
	2.3.2.3.5.2	pots_comm_trav_real	727
	2.3.2.3.5.3	pots_comm_elev_real.....	727
	2.3.2.3.5.4	pots_cupola_real.....	728
	2.3.2.3.5.5	pots_gunn_trav_real	728
	2.3.2.3.5.6	pots_gunn_elev_real.....	729
	2.3.2.3.5.7	pots_steer_bar_real.....	729
	2.3.2.3.5.8	pots_throttle_real.....	730
	2.3.2.3.5.9	pots_service_brake_real	730

	2.3.2.3.5.10	pots_accelerator_real.....	731
	2.3.2.3.6	m2_slope.c.....	732
	2.3.2.3.6.1	slope_simul.....	732
	2.3.2.3.6.2	slope_get_cos_hull_slope.....	733
2.3.3	M2 Weapons		734
	2.3.3.1	m2_bcs.c.....	736
	2.3.3.1.1	bcs_init	737
	2.3.3.1.2	bcs_simul.....	737
	2.3.3.1.3	bcs_ammo_index_hei_25.....	737
	2.3.3.1.4	bcs_ammo_index_apds_25	738
	2.3.3.1.5	bcs_ammo_index_no_round	738
	2.3.3.1.6	bcs_range_is	738
	2.3.3.1.7	bcs_set_ballistics_computer.....	739
	2.3.3.1.8	bcs_get_super_elevation	739
	2.3.3.1.9	bcs_get_range.....	739
	2.3.3.1.10	bcs_get_time_of_flight.....	739
	2.3.3.1.11	bcs_get_ammo_type_indexed	740
	2.3.3.1.12	bcs_get_range_str.....	740
	2.3.3.1.13	bcs_turn_computer_on	740
	2.3.3.1.14	bcs_turn_computer_off	740
	2.3.3.1.15	bcs_str_null	741
	2.3.3.2	m2_weapons.c	741
	2.3.3.2.1	weapons_missile_is_launched	743
	2.3.3.2.2	tow_fired_check	743
	2.3.3.2.3	weapons_fire_round.....	744
	2.3.3.2.4	weapons_fire	745
	2.3.3.2.5	weapons_simul.....	746
	2.3.3.2.6	weapons_init.....	746
	2.3.3.2.7	weapons_set_low_fire_rate.....	746
	2.3.3.2.8	weapons_trigger_is_released	747
	2.3.3.2.9	weapons_cut_any_tow_wires.....	747
	2.3.3.2.10	weapons_set_high_fire_rate.....	747
	2.3.3.2.11	weapons_set_single_shot_ mode	747
	2.3.3.2.12	weapons_trigger_is_pulled.....	747
	2.3.3.2.13	weapons_trigger_status	747
	2.3.3.2.14	weapons_shot_misfired	748
	2.3.3.2.15	weapons_break_tow_ launcher	748
	2.3.3.2.16	weapons_repair_tow_ launcher	748
	2.3.3.2.17	weapons_misfire_corrected.....	748
	2.3.3.2.18	weapons_vehicle_rolled	748

	2.3.3.2.19	weapons_vehicle_unrolled	748
	2.3.3.2.20	weapons_download_ ballistics_ tables.....	748
	2.3.3.2.21	weapons_keybrd_fire	749
2.3.4	M2 Failures		750
	2.3.4.1	m2_failure.c.....	751
	2.3.4.1.1	fail_init	752
	2.3.4.1.2	failure_collision_damages.....	753
	2.3.4.1.3	failure_check_cat_kill	754
	2.3.4.1.4	failure_check_indir_fire_ damages.....	755
	2.3.4.2	m2_repair.c.....	756
	2.3.4.2.1	repair_request	757
	2.3.4.2.2	repair_simul.....	757
	2.3.4.2.3	repair_init	758
	2.3.4.2.4	clear_repair_vehicles.....	758
	2.3.4.2.5	repair_near_repair	758
	2.3.4.2.6	send_feed_me_packets_ repair_ vehicles.....	758
	2.3.4.2.7	repair_quiet_state	759
	2.3.4.2.8	repair_request_state.....	759
	2.3.4.2.9	print_repair_status.....	760
2.3.5	M.2 Munitions Management.....		761
	2.3.5.1	m2_ammo.c.....	762
	2.3.5.1.1	ammo_init.....	764
	2.3.5.1.2	ammo_simul	764
	2.3.5.1.3	ammo_init_ammo_supply.....	765
	2.3.5.1.4	ammo_get_apds_can_ quantity.....	765
	2.3.5.1.5	ammo_get_apds_can_ammo_ boxes.....	765
	2.3.5.1.6	ammo_get_hei_can_quantity	766
	2.3.5.1.7	ammo_get_hei_can_ammo_ boxes.....	766
	2.3.5.1.8	ammo_get_apds_stowed_ quantity.....	766
	2.3.5.1.9	ammo_get_hei_stowed_ quantity.....	766
	2.3.5.1.10	ammo_get_tow_stowed_ quantity.....	766
	2.3.5.1.11	ammo_get_dragon_stowed_ quantity.....	767
	2.3.5.1.12	ammo_get_missile1_val.....	767

2.3.5.1.13	ammo_get_missile2_val.....	767
2.3.5.1.14	ammo_get_m3_configuration_val	767
2.3.5.1.15	ammo_hei_can_hei_on.....	767
2.3.5.1.16	ammo_hei_can_hei_off.....	768
2.3.5.1.17	ammo_apds_can_hei_on	768
2.3.5.1.18	ammo_apds_can_hei_off	768
2.3.5.1.19	ammo_mgmt_receive_pushed.....	768
2.3.5.1.20	ammo_mgmt_send_pushed.....	768
2.3.5.1.21	ammo_mgmt_internal_pushed.....	769
2.3.5.1.22	ammo_mgmt_hei_pushed	769
2.3.5.1.23	ammo_mgmt_apds_pushed.....	769
2.3.5.1.24	ammo_mgmt_tow_pushed	769
2.3.5.1.25	ammo_mgmt_dragon_pushed	770
2.3.5.1.26	ammo_round_selected_status.....	770
2.3.5.1.27	ammo_round_loaded_status.....	770
2.3.5.1.28	ammo_round_indexed_status.....	770
2.3.5.1.29	ammo_reversed_check.....	771
2.3.5.1.30	ammo_reversed_status	771
2.3.5.1.31	ammo_indexed_check.....	771
2.3.5.1.32	ammo_ap_ss_pushed.....	772
2.3.5.1.33	ammo_he_ss_pushed.....	772
2.3.5.1.34	ammo_ap_lo_pushed.....	773
2.3.5.1.35	ammo_he_lo_pushed.....	773
2.3.5.1.36	ammo_ap_hi_pushed.....	774
2.3.5.1.37	ammo_he_hi_pushed.....	774
2.3.5.1.38	ammo_tow_select_pushed	775
2.3.5.1.39	ammo_tow_test_pushed	775
2.3.5.1.40	ammo_missile1_pushed	775
2.3.5.1.41	ammo_missile2_pushed	776
2.3.5.1.42	ammo_tow_launcher_on	776
2.3.5.1.43	ammo_tow_launcher_off	776
2.3.5.1.44	ammo_gps_mag_12x	776
2.3.5.1.45	ammo_gps_mag_4x	777
2.3.5.1.46	ammo_tow_test_check	777
2.3.5.1.47	ammo_tow_test_start	777
2.3.5.1.48	ammo_tow_test_stop.....	777
2.3.5.1.49	ammo_low_ammo_check.....	778
2.3.5.1.50	ammo_low_ammo_pushed.....	778
2.3.5.1.51	ammo_low_ammo_ready_to_fire	778
2.3.5.1.52	ammo_turret_power_off.....	779
2.3.5.1.53	ammo_bolt_position_status.....	779

2.3.5.1.54	ammo_misfire_lock_status.....	779
2.3.5.1.55	ammo_weapon_removed	780
2.3.5.1.56	ammo_weapon_is_fired	780
2.3.5.1.57	ammo_weapon_is_misfired	780
2.3.5.1.58	ammo_misfire_corrected.....	781
2.3.5.1.59	ammo_misfire_pushed	781
2.3.5.1.60	ammo_restore_ammo	781
2.3.5.1.61	ammo_get_missile_loaded	781
2.3.5.1.62	ammo_get_apds_can_first_	
	round.....	782
2.3.5.1.63	ammo_get_hei_can_first_	
	round.....	782
2.3.5.1.64	ammo_remove_apds_can_	
	round.....	782
2.3.5.1.65	ammo_remove_hei_can_	
	round.....	782
2.3.5.1.66	ammo_get_apds_can_box	783
2.3.5.1.67	ammo_get_hei_can_box.....	783
2.3.5.1.68	ammo_print_ammo_variables.....	783
2.3.5.1.69	ammo_ready_to_internal_	
	resupply	784
2.3.5.1.70	ammo_ready_to_external_	
	resupply	786
2.3.5.1.71	ammo_ready_to_external_	
	send.....	787
2.3.5.1.72	ammo_supply_empty_	
	stowage.....	788
2.3.5.1.73	ammo_start_internal_	
	resupply	788
2.3.5.1.74	ammo_start_external_	
	resupply	789
2.3.5.1.75	ammo_start_external_send.....	789
2.3.5.1.76	ammo_stop_resupply	790
2.3.5.1.77	ammo_hei_can_enough_	
	room	790
2.3.5.1.78	ammo_apds_can_enough_	
	room	790
2.3.5.1.79	ammo_25mm_stowage_	
	enough_room.....	791
2.3.5.1.80	ammo_tow_tubes_enough_	
	room	791
2.3.5.1.81	ammo_tow_stowage_enough_	
	room	791
2.3.5.1.82	ammo_dragon_stowage_	
	enough_room.....	791

	2.3.5.1.83	ammo_hei_stowage_enough_	
		supply	792
	2.3.5.1.84	ammo_apds_stowage_	
		enough_supply	792
	2.3.5.1.85	ammo_tow_stowage_enough_	
		supply	792
	2.3.5.1.86	ammo_turret_no_power_off.....	792
	2.3.5.1.87	ammo_internal_resupply_	
		start_check.....	793
	2.3.5.1.88	ammo_internal_resupply_	
		abort_check	793
	2.3.5.1.89	ammo_resupply_timeout_	
		check.....	794
	2.3.5.1.90	ammo_rounds_on_board_	
		check.....	794
	2.3.5.1.91	ammo_resupply_sent.....	795
	2.3.5.1.92	ammo_decide_round_type	795
2.3.5.2	m2_fuelsys.c.....		796
	2.3.5.2.1	fuel_init_tanks.....	797
	2.3.5.2.2	fuel_init	797
	2.3.5.2.3	fuel_simul.....	797
	2.3.5.2.4	fuel_top_tank_not_empty.....	798
	2.3.5.2.5	fuel_set_flow	798
	2.3.5.2.6	fuel_xfer_fuel	798
	2.3.5.2.7	fuel_engine_accessory_on	799
	2.3.5.2.8	fuel_engine_accessory_off.....	799
	2.3.5.2.9	fuel_level_bottom.....	799
	2.3.5.2.10	fuel_level_top.....	799
	2.3.5.2.11	fuel_supply_full	800
	2.3.5.2.12	fuel_decide_resupply_	
		quantity.....	800
	2.3.5.2.13	fuel_start_external_resupply	801
	2.3.5.2.14	fuel_stop_resupply	801
	2.3.5.2.15	fuel_resupply_tank	801
	2.3.5.2.16	print_fuel_variables.....	801
	2.3.5.2.17	fuel_on.....	802
	2.3.5.2.18	fuel_off	802
	2.3.5.2.19	fuel_transfer_pump_failed	802
	2.3.5.2.20	fuel_repair_transfer_pump	802
2.3.5.3	m2_resupp.c		802
	2.3.5.3.1	clear_ammo_carriers	803
	2.3.5.3.2	clear_fuel_carriers.....	804
	2.3.5.3.3	clear_ammo_receivers.....	804
	2.3.5.3.4	print_resupply_status	804

2.3.5.3.5	send_feed_me_packets_	
	ammo_carriers.....	804
2.3.5.3.6	send_feed_me_packets_fuel_	
	carriers.....	804
2.3.5.3.7	resupply_near_ammo_carrier.....	805
2.3.5.3.8	resupply_near_fuel_carrier.....	805
2.3.5.3.9	resupply_near_ammo_	
	receiver.....	805
2.3.5.3.10	resupply_ammo_received.....	805
2.3.5.3.11	resupply_fuel_received	806
2.3.5.3.12	resupply_offer_packet.....	807
2.3.5.3.13	resupply_thank_you_packet.....	808
2.3.5.3.14	resupply_feed_me_packet.....	809
2.3.5.3.15	resupply_gating_conditions	810
2.3.5.3.16	ammo_receive_quiet_state	811
2.3.5.3.17	fuel_receive_quiet_state.....	812
2.3.5.3.18	ammo_send_quiet_state	812
2.3.5.3.19	ammo_receive_request_state	813
2.3.5.3.20	fuel_receive_request_state	814
2.3.5.3.21	ammo_send_waiting_state	815
2.3.5.3.22	ammo_receive_loading_state	816
2.3.5.3.23	fuel_receive_loading_state.....	817
2.3.5.3.24	ammo_send_servicing_state.....	818
2.3.5.3.25	ammo_resupply_receive_	
	simul.....	818
2.3.5.3.26	fuel_resupply_receive_simul	819
2.3.5.3.27	ammo_resupply_send_simul.....	819
2.3.5.3.28	resupply_init.....	819
2.3.5.3.29	resupply_simul	820
2.3.5.3.29	service_check_vehicle_type.....	820
2.3.5.3.30	resupply_stop_ammo_	
	resupply	821
2.3.5.3.31	resupply_stop_fuel_resupply	821
2.3.5.3.32	resupply_offer_canceled	821
2.3.5.3.33	resupply_request_canceled.....	821
2.3.5.3.34	vehicle_is_close	822
2.3.5.3.35	keybrd_ammo_carriers_near_	
	here.....	822
2.3.6	M2 Vehicle Model	823
2.3.6.1	Internal Kinematics	824
2.3.6.1.1	m2_turret.c	824
2.3.6.1.1.1	turret_init.....	825
2.3.6.1.1.2	turret_simul	826
2.3.6.1.1.3	turret_move	826

2.3.6.1.1.4	turret_get_turret_slew_rate	827
2.3.6.1.1.5	turret_get_gun_elev_rate.....	827
2.3.6.1.1.6	turret_handles_values.....	827
2.3.6.1.1.7	turret_calc_turret_slew.....	828
2.3.6.1.1.8	calc_slew_from_handle.....	828
2.3.6.1.1.9	turret_calc_gun_elev	829
2.3.6.1.1.10	calc_elev_from_handle	829
2.3.6.1.1.11	turret_gyros_simul	830
2.3.6.1.1.12	turret_stab_on.....	830
2.3.6.1.1.13	turret_stab_off.....	830
2.3.6.1.1.14	turret_gyros_spool_up.....	830
2.3.6.1.1.15	turret_gyros_spool_down.....	831
2.3.6.1.1.16	turret_gyros_status	831
2.3.6.1.1.17	turret_break_elevation_drive	831
2.3.6.1.1.18	turret_repair_elevation_drive	831
2.3.6.1.1.19	turret_break_stab_system.....	831
2.3.6.1.1.20	turret_repair_stab_system	831
2.3.6.1.1.21	turret_break_mount_interface	831
2.3.6.1.1.22	turret_repair_mount_interface.....	831
2.3.6.1.1.23	turret_break_traverse_drive	831
2.3.6.1.1.24	turret_repair_traverse_drive	832
2.3.6.1.1.25	turret_collision_detected	832
2.3.6.1.1.25	make_sound_of_no_slewing.....	832
2.3.6.1.1.26	make_sound_of_no_elevating.....	832
2.3.6.1.1.27	make_sound_of_no_turret_	
	noise	833
2.3.6.1.1.28	turret_get_gun_to_world.....	833
2.3.6.1.1.29	turret_tow_movement_off.....	833
2.3.6.1.1.30	turret_tow_movement_on	833
2.3.6.1.1.31	turret_set_super_elevation	834
2.3.6.1.2	m2_cupola.c	835
2.3.6.1.2.1	cupola_get_cws_cos_and_sin	835
2.3.6.1.2.2	cupola_get_real_cws_cos_	
	and_sin	836
2.3.6.1.2.3	convert_disp_to_angle	836
2.3.6.1.2.4	cupola_cws_new_value.....	836
2.3.6.1.2.5	cupola_simul	837
2.3.6.1.2.6	cupola_init.....	837
2.3.6.1.3	m2_ramp.c.....	838
2.3.6.1.3.1	ramp_init_ramp_down	838
2.3.6.1.3.2	ramp_simul.....	839
2.3.6.1.3.3	ramp_up.....	839
2.3.6.1.3.4	ramp_down.....	839

2.3.6.1.3.5	ramp_idle.....	839
2.3.6.1.3.6	ramp_get_val.....	840
2.3.6.1.3.7	ramp_down_status.....	840
2.3.6.1.4	m2_launcher.c	841
2.3.6.1.4.1	launcher_init_launcher_up	841
2.3.6.1.4.2	launcher_simul	842
2.3.6.1.4.3	launcher_up	842
2.3.6.1.4.4	launcher_down	842
2.3.6.1.4.5	launcher_idle	843
2.3.6.1.4.6	launcher_get_val	843
2.3.6.1.4.7	launcher_up_status	843
2.3.6.2	M2 Propulsion Simulation	844
2.3.6.2.1	m2_pttrain.c.....	844
2.3.6.2.1.1	powertrain_init	844
2.3.6.2.1.2	powertrain_simul.....	844
2.3.6.2.2	m2_trans.c	845
2.3.6.2.2.1	transmission_break_ transmission.....	845
2.3.6.2.2.2	transmission_replace_ transmission.....	846
2.3.6.2.2.3	transmission_init	846
2.3.6.2.2.4	transmission_simul.....	847
2.3.6.2.2.5	fit_T_init.....	847
2.3.6.2.2.6	fit_Q_init	847
2.3.6.2.2.7	fit_Q	848
2.3.6.2.2.8	fit_T.....	849
2.3.6.2.2.9	transmission_load_torque.....	849
2.3.6.2.2.10	transmission_torque_left	849
2.3.6.2.2.11	transmission_torque_right.....	850
2.3.6.2.2.12	transmission_oil_leak.....	850
2.3.6.2.2.13	transmission_repair_oil_leak	850
2.3.6.2.2.14	Debugging tools	850
2.3.6.2.3	m2_cntrlr.c	851
2.3.6.2.3.1	controller_init.....	852
2.3.6.2.3.2	controller_simul	852
2.3.6.2.3.3	fit_speed_ref	853
2.3.6.2.3.4	shift_check	853
2.3.6.2.3.5	stroke_calc.....	854
2.3.6.2.3.6	controller_stroke_left	854
2.3.6.2.3.7	controller_stroke_right	854
2.3.6.2.3.8	controller_gear	855
2.3.6.2.3.9	controller_neutral	855
2.3.6.2.3.10	controller_pivot.....	855

2.3.6.2.3.11	controller_drive	855
2.3.6.2.3.12	controller_low	855
2.3.6.2.3.13	controller_reverse.....	855
2.3.6.2.3.14	controller_start	855
2.3.6.2.3.15	controller_set_throttle	856
2.3.6.2.3.16	controller_set_steering_bar	856
2.3.6.2.3.17	cntrlr_dump	856
2.3.6.2.3.18	cntrlr_banner	856
2.3.6.2.3.19	cntrlr_data_title	856
2.3.6.2.3.20	cntrlr_data_banner.....	856
2.3.6.2.3.21	cntrlr_data_dump	856
2.3.6.2.4	m2_dtrain.c.....	857
2.3.6.2.4.1	check_for_thrown_track.....	859
2.3.6.2.4.2	drivetrain_simul	860
2.3.6.2.4.3	rotational_friction_factor	860
2.3.6.2.4.4	compute_traction_force.....	861
2.3.6.2.4.5	check_for_slip	861
2.3.6.2.4.6	check_forward_collision	862
2.3.6.2.4.7	check_side_collision	862
2.3.6.2.4.8	drivetrain_get_vehicle_speed.....	863
2.3.6.2.4.9	drivetrain_left_omega	863
2.3.6.2.4.10	drivetrain_right_omega	863
2.3.6.2.4.11	drivetrain_set_brake	863
2.3.6.2.4.12	drivetrain_parking_brake_set.....	864
2.3.6.2.4.13	drivetrain_parking_brake_	
	release.....	864
2.3.6.2.4.14	drivetrain_service_brake_	
	failure	864
2.3.6.2.4.15	drivetrain_repair_service_	
	brake	864
2.3.6.2.4.16	drivetrain_parking_brake_	
	failure	864
2.3.6.2.4.17	drivetrain_repair_parking_	
	brake	864
2.3.6.2.4.18	drivetrain_throw_right_track	865
2.3.6.2.4.19	drivetrain_throw_left_track.....	865
2.3.6.2.4.20	drivetrain_repair_thrown_	
	tracks	865
2.3.6.2.4.21	dump_drivetrain_state.....	865
2.3.6.2.4.22	drivetrain_data_title	865
2.3.6.2.4.23	drivetrain_data_banner.....	865
2.3.6.2.4.24	drivetrain_data_dump.....	865
2.3.6.2.4.25	drivetrain_banner	866
2.3.6.2.4.26	drivetrain_dump	866

2.3.6.2.4.27	drivetrain_init	866
2.3.6.2.5	m2_engine.c	867
2.3.6.2.5.1	engine_simul	869
2.3.6.2.5.2	engine_start	870
2.3.6.2.5.3	engine_out_of_start	870
2.3.6.2.5.4	engine_accessory_on	870
2.3.6.2.5.5	engine_accessory_off	871
2.3.6.2.5.6	engine_fail	871
2.3.6.2.5.7	engine_fix	871
2.3.6.2.5.8	engine_starter_fail	871
2.3.6.2.5.9	engine_starter_fix	871
2.3.6.2.5.10	engine_running	871
2.3.6.2.5.11	engine_speed	872
2.3.6.2.5.12	engine_get_speed	872
2.3.6.2.5.13	engine_rpm	872
2.3.6.2.5.14	engine_get_max_power	872
2.3.6.2.5.15	engine_set_throttle	873
2.3.6.2.5.16	engine_run	873
2.3.6.2.5.17	engine_crank	873
2.3.6.2.5.18	engine_off	873
2.3.6.2.5.19	fit_engine_torque	874
2.3.6.2.5.20	engine_dump	874
2.3.6.2.5.21	engine_banner	875
2.3.6.2.5.22	engine_data_title	875
2.3.6.2.5.23	engine_banner	875
2.3.6.2.5.24	engine_data_dump	875
2.3.6.2.5.25	engine_init	875
2.3.6.2.6	m2_engfail.c	876
2.3.6.2.6.1	engine_coolant_leak	877
2.3.6.2.6.2	engine_coolant_normal	877
2.3.6.2.6.3	engine_clog_fuel_filter	877
2.3.6.2.6.4	engine_fix_fuel_filter	878
2.3.6.2.6.5	engine_clog_air_filter	878
2.3.6.2.6.6	engine_fix_air_filter	878
2.3.6.2.6.7	engine_oil_leak	878
2.3.6.2.6.8	engine_oil_normal	879
2.3.6.2.6.9	engine_fail_starter	879
2.3.6.2.6.10	engine_fix_starter	879
2.3.6.2.6.11	engine_failure_update	879
2.3.6.2.6.12	engine_init_power	880
2.3.6.2.6.13	engine_power_loss	880
2.3.6.2.6.14	check_starter_failure	880
2.3.6.2.6.15	check_engine_failure	881

2.3.6.3	Vehicle Subsystems.....	881
2.3.6.3.1	m2_electsys.c	881
2.3.6.3.1.1	electsys_charge_battery	885
2.3.6.3.1.2	electsys_discharge_hull_ battery.....	885
2.3.6.3.1.3	electsys_discharge_turret_ backup_battery	886
2.3.6.3.1.4	electsys_rads_to_volts.....	886
2.3.6.3.1.5	electsys_handle_leaky_hull_ battery.....	887
2.3.6.3.1.6	electsys_handle_leaky_turret_ backup_battery	887
2.3.6.3.1.7	electsys_turret_backup_ power_request	888
2.3.6.3.1.8	electsys_simul	888
2.3.6.3.1.9	electsys_hull_dead	889
2.3.6.3.1.10	electsys_turret_dead	889
2.3.6.3.1.11	electsys_dead.....	889
2.3.6.3.1.12	electsys_drive_malfunction_ status.....	889
2.3.6.3.1.13	electsys_set_turret_drive_ status.....	890
2.3.6.3.1.14	electsys_25mm_gun_ malfunction_status	890
2.3.6.3.1.15	electsys_set_25mm_gun_ malfunction_status	890
2.3.6.3.1.16	electsys_tow_circuit_ open_status.....	890
2.3.6.3.1.17	electsys_set_tow_circuit_ open_status.....	891
2.3.6.3.1.18	electsys_hull_power_request	891
2.3.6.3.1.19	electsys_turret_power_ request	891
2.3.6.3.1.20	electsys_engine_start_request	892
2.3.6.3.1.21	electsys_tow_request.....	892
2.3.6.3.1.22	electsys_turret_elevation_ request	893
2.3.6.3.1.23	electsys_turret_traverse_ request	894
2.3.6.3.1.24	electsys_25mm_gun_request	894
2.3.6.3.1.25	electsys_fuel_xfer_pump_ request	895
2.3.6.3.1.26	electsys_get_hull_battery_ voltage.....	895
2.3.6.3.1.27	electsys_get_turret_backup_ battery_voltage.....	895

2.3.6.3.1.28	electsys_replace_generator.....	895
2.3.6.3.1.29	electsys_generator_failure.....	895
2.3.6.3.1.30	electsys_replace_hull_battery	896
2.3.6.3.1.31	electsys_replace_turret_ backup_battery	896
2.3.6.3.1.32	electsys_turret_power_off.....	896
2.3.6.3.1.33	electsys_hull_power_off	896
2.3.6.3.1.34	print_electsys_variables	897
2.3.6.3.1.35	electsys_reborn.....	897
2.3.6.3.1.36	electsys_turret_reborn	897
2.3.6.3.1.37	electsys_hull_reborn.....	897
2.3.6.3.1.38	electsys_init_batteries	898
2.3.6.3.1.39	electsys_voltmeter_disabled.....	898
2.3.6.3.1.40	electsys_init.....	898
2.3.6.3.2	m2_vision.c	899
2.3.6.3.2.1	vision_get_sky_color	901
2.3.6.3.2.2	vision_toggle_sky_color	901
2.3.6.3.2.3	cig_gps_mag_12x.....	901
2.3.6.3.2.4	cig_gps_mag_4x.....	901
2.3.6.3.2.5	vision_cmdrs_pitch_up	901
2.3.6.3.2.6	vision_cmdrs_pitch_ahead.....	902
2.3.6.3.2.7	vision_cmdrs_pitch_down	902
2.3.6.3.2.8	vision_restore_all_blocks.....	903
2.3.6.3.2.9	vision_break_all_blocks.....	903
2.3.6.3.2.10	vision_break_isu.....	903
2.3.6.3.2.11	vision_break_isu_ext.....	904
2.3.6.3.2.12	vision_break_driver_blocks	904
2.3.6.3.2.13	vision_break_cmdrs_blocks.....	904
2.3.6.3.2.14	vision_restore_isu.....	905
2.3.6.3.2.15	vision_restore_isu_ext.....	905
2.3.6.3.2.16	vision_restore_driver_blocks	905
2.3.6.3.2.17	vision_restore_cmdrs_blocks.....	906
2.3.6.3.2.18	vision_break_gunners_block.....	906
2.3.6.3.2.19	vision_restore_gunners_block.....	906
2.3.6.3.2.20	vision_gunner_brow_pad_on	907
2.3.6.3.2.21	vision_gunner_brow_pad_off	907
2.3.6.3.2.22	vision_commander_brow_ pad_on	907
2.3.6.3.2.23	vision_commander_brow_ pad_off	908
2.3.6.3.2.24	print_br_values.....	908
2.3.6.3.2.25	get_cmdr_state	908
2.3.6.3.2.26	get_gunner_state.....	909

	2.3.6.3.2.27	get_brow_pad_status.....	909
	2.3.6.3.2.28	vision_init.....	909
	2.3.6.3.3	m2_isu.c	910
	2.3.6.3.3.1	isu_init.....	910
	2.3.6.3.3.2	isu_simul	911
	2.3.6.3.3.3	isu_gps_mag_12x.....	911
	2.3.6.3.3.4	isu_gps_mag_4x.....	911
	2.3.6.3.3.5	isu_round_select_25mm.....	911
	2.3.6.3.3.6	isu_round_select_no_round	911
	2.3.6.3.3.7	isu_round_select_tow.....	911
2.3.7	Network Interactions.....		912
	2.3.7.1	m2_network.c	913
	2.3.7.1.1	send_equipment_status.....	914
	2.3.7.1.2	fill_vehicle_spec_status	915
	2.3.7.1.3	fill_vehicle_spec_appearance.....	916
	2.3.7.1.4	network_process_activation_	
		parameters	916
	2.3.7.1.5	app_init.....	917
	2.3.7.1.6	veh_spec_activate_time	917
	2.3.7.2	m2_dust.c	918
	2.3.7.2.1	tracks_get_dust_cloud.....	918
2.4	STEALTH VEHICLE.....		919
2.4.1	Top Level Stealth Simulation Software		920
	2.4.1.1	kato_main.c	920
	2.4.1.1.1	print_help	921
	2.4.1.1.2	print_veh_logo	921
	2.4.1.1.3	veh_spec_startup	922
	2.4.1.1.4	veh_spec_idle	922
	2.4.1.1.5	veh_spec_init.....	923
	2.4.1.1.6	veh_spec_simulate	923
	2.4.1.1.7	veh_spec_stop	924
	2.4.1.1.8	veh_spec_exit.....	924
	2.4.1.1.9	main.....	925
	2.4.1.1.10	reconstitute_vehicle.....	927
2.4.2	Stealth Controls/Switchology.....		928
	2.4.2.1	kato_ctl_fsm.c	928
	2.4.2.1.1	controls_fsm_init.....	928
	2.4.2.1.2	controls_simul	929
	2.4.2.1.3	controls_power_status.....	929
	2.4.2.1.4	controls_break_controls	929
	2.4.2.1.5	controls_restore_controls	930
	2.4.2.1.6	controls_failure_status	930
	2.4.2.1.7	controls_edges_clear	930

	2.4.2.1.8	controls_edge_init	930
	2.4.2.1.9	controls_lamp_init	930
	2.4.2.1.10	controls_sim_next_state	931
	2.4.2.2	kato_ctl_nls.c	931
	2.4.2.3	kato_ctl_sim.c	931
	2.4.2.4	kato_pots.c	931
	2.4.2.5	kato_meter.c	931
2.4.3	Stealth Failures		932
	2.4.3.1	kato_failure.c	932
2.4.4	Munitions Management		932
	2.4.4.1	kato_ammo.c	932
	2.4.4.2	kato_resupp.c	932
2.4.5	Stealth Vehicle Model		932
	2.4.5.1	kato_vision.c	932
	2.4.5.2	kato_rotate.c	932
2.4.6	Network Interactions		933
	2.4.6.1	kato_network.c	933
	2.4.6.2	kato_cmcflt.c	933
	2.4.6.3	kato_periph.c	933
	2.4.6.4	kato_simul.c	933
	2.4.6.5	kato_stubs.c	933
2.4.7	Attach Capability		934
	2.4.7.1	kato_attach.c	934
	2.4.7.2	kato_gunmnt.c	934
2.4.8	Motion		934
	2.4.8.1	kato_control.c	934
	2.4.8.2	kato_state.c	934
2.5	VEHICLE LIBRARIES		935
2.5.1	libmain		937
	2.5.1.1	main.c	937
	2.5.1.1.1	enter_gracefully	938
	2.5.1.1.2	exit_gracefully	938
	2.5.1.1.3	activate_simulation	939
	2.5.1.1.4	deactivate_simulation	939
	2.5.1.1.5	sim_state_startup	939
	2.5.1.1.6	sim_state_idle	939
	2.5.1.1.7	sim_state_siminit	939
	2.5.1.1.8	sim_state_simulate	939
	2.5.1.1.9	sim_state_simstop	939
	2.5.1.1.10	sim_state_simexit	939
	2.5.1.1.11	sim_state_simulating	940
	2.5.1.1.12	sim_state_sounds_denied	940
	2.5.1.1.13	simulation_state_machine	940

2.5.1.2	read_pars.c.....	943
2.5.1.2.1	main_read_pars_file.....	944
2.5.1.2.2	get_vconfig_file1	944
2.5.1.2.3	get_vconfig_file2	944
2.5.1.2.4	get_asid_map_file	945
2.5.1.2.5	get_veh_map_file.....	945
2.5.1.2.6	get_ammo_map_file.....	945
2.5.1.2.7	get_sdamage_file.....	945
2.5.1.2.8	get_thresh_file.....	946
2.5.1.2.9	get_idle_filter_file.....	946
2.5.1.2.10	get_priority_list_file.....	946
2.5.1.2.11	get_register_file.....	946
2.5.1.2.12	get_device_file	947
2.5.1.2.13	get_calib_file.....	947
2.5.1.2.14	get_sim_filter_file.....	947
2.5.1.2.15	get_default_db_name	947
2.5.1.2.16	get_default_db_version	948
2.5.1.2.17	get_ded_override.....	948
2.5.1.2.18	get_db_override.....	948
2.5.1.2.19	get_constants_file.....	948
2.5.1.2.20	print_pars_files.....	949
2.5.2	libball.....	950
2.5.2.1	ball_calc.c.....	950
2.5.2.1.1	ballistics_calc_time	951
2.5.2.1.2	ballistics_calc_se.....	952
2.5.2.2	ball_fire.c.....	953
2.5.2.2.1	ballistics_fire_a_round	953
2.5.2.3	ball_load.c	954
2.5.2.3.1	ballistics_load_trajectory_file	955
2.5.2.3.3	ballistics_load_parameter_file	956
2.5.2.4	ball_orient.c.....	957
2.5.2.4.1	ballistics_cal_azm_elev.....	957
2.5.3	libmissile	958
2.5.3.1	fuze_prox.c.....	958
2.5.3.1.1	missile_fuze_prox_init.....	958
2.5.3.1.2	missile_fuze_prox	959
2.5.3.1.3	missile_fuze_prox_stop.....	961
2.5.3.1.4	get_prox.....	962
2.5.3.1.5	free_prox	962
2.5.3.1.6	f2d_mat_transpose	962
2.5.3.1.7	dfd_vec_sub	963
2.5.3.1.8	f2d_vec_scale.....	963
2.5.3.2	libmiss_dfn.h.....	963

2.5.3.3	libmiss_loc.h	963
2.5.3.4	libmissile.h	963
2.5.3.5	miss_adat.c	964
2.5.3.2.1	missile_adat_init.....	965
2.5.3.2.2	missile_adat_fire	965
2.5.3.2.3	missile_adat_fly_missiles.....	967
2.5.3.2.4	missile_adat_fly	968
2.5.3.2.4	missile_adat_reset_missiles	969
2.5.3.2.5	missile_adat_stop	969
2.5.3.6	miss_adat.h	969
2.5.3.7	miss_hellfr.c	970
2.5.3.7.1	missile_hellfire_init.....	971
2.5.3.7.2	missile_hellfire_fire	971
2.5.3.7.3	missile_hellfire_fly.....	972
2.5.3.7.4	missile_hellfire_stop	972
2.5.3.8	miss_hellfr.h	973
2.5.3.9	miss_maverick.c	973
2.5.3.9.1	missile_maverick_init	974
2.5.3.9.2	missile_maverick_ready.....	975
2.5.3.9.3	missile_maverick_pre_launch.....	975
2.5.3.9.4	missile_maverick_fire	976
2.5.3.9.5	missile_maverick_fly_ missiles.....	977
2.5.3.9.6	missile_maverick_fly	978
2.5.3.9.7	missile_maverick_stop	979
2.5.3.10	miss_maverick.h.....	979
2.5.3.11	miss_stinger.c	979
2.5.3.11.1	missile_stinger_init	980
2.5.3.11.2	missile_stinger_ready.....	981
2.5.3.11.3	missile_stinger_pre_launch.....	981
2.5.3.11.4	missile_stinger_fire	982
2.5.3.11.5	missile_stinger_fly_missiles	983
2.5.3.11.6	missile_stinger_fly	984
2.5.3.11.7	missile_stinger_stop	985
2.5.3.12	miss_stinger.h.....	985
2.5.3.13	miss_tow.c	985
2.5.3.13.1	missile_tow_init	986
2.5.3.13.2	missile_tow_fire	987
2.5.3.13.3	missile_tow_fly	988
2.5.3.13.4	missile_tow_stop.....	989
2.5.3.13.5	missile_tow_cut_wire.....	989
2.5.3.14	miss_tow.h.....	989
2.5.3.15	targ_agm.c	989

	2.5.3.15.1	missile_target_agm.....	990
	2.5.3.15.2	missile_agm_seek.....	992
2.5.3.16	targ_ground.c.....		993
	2.5.3.16.1	missile_target_ground	993
2.5.3.17	targ_intcpt.c		993
	2.5.3.17.1	missile_target_intercept_pre_ burnout	994
	2.5.3.17.2	missile_target_intercept	995
	2.5.3.17.3	missile_target_intercept_find_ poly.....	997
2.5.3.18	targ_lev_los.c		998
	2.5.3.18.1	missile_target_level_los.....	998
2.5.3.19	targ_los.c		1000
	2.5.3.19.1	missile_target_los.....	1000
2.5.3.20	targ_losbias.c.....		1001
	2.5.3.20.1	missile_target_los_bias	1001
2.5.3.21	targ_point.c.....		1003
	2.5.3.21.1	missile_target_point	1003
2.5.3.22	targ_pursuit.c.....		1004
	2.5.3.22.1	missile_target_pursuit	1004
2.5.3.23	targ_unguide.c.....		1005
	2.5.3.23.1	missile_target_unguided.....	1005
2.5.3.24	util_comm.c.....		1006
	2.5.3.24.1	missile_util_comm_init.....	1007
	2.5.3.24.2	missile_util_comm_fire_ missile.....	1008
	2.5.3.24.3	missile_util_comm_fly_ missile.....	1010
	2.5.3.24.4	missile_util_comm_ intersected_poly	1011
	2.5.3.24.5	missile_util_comm_ intersected_model.....	1011
	2.5.3.24.6	missile_util_comm_fuze_ detonate	1012
	2.5.3.24.7	missile_util_comm_stop_ missile.....	1013
	2.5.3.24.8	missile_util_comm_check_ intersection	1013
	2.5.3.24.9	missile_util_comm_check_ detonate	1014
2.5.3.25	util_eval.c		1015
	2.5.3.25.1	missile_util_eval_poly	1015
	2.5.3.25.2	missile_util_eval_cos_coeff	1016
	2.5.3.25.3	missile_util_eval_newton_ raphson	1017

2.5.3.26	util_flyout.c	1019
2.5.3.26.1	missile_util_flyout.....	1020
2.5.3.27	util_init.c	1022
2.5.3.27.1	missile_util_init.....	1022
2.5.4	libfail	1023
2.5.4.1	c_chk_dam.c.....	1024
2.5.4.1.1	cfail_check_damages	1024
2.5.4.2	c_debug.c.....	1026
2.5.4.2.1	cfail_debug_on	1026
2.5.4.3	c_dir_fire.c	1027
2.5.4.3.1	cfail_dir_fire_damages.....	1028
2.5.4.3.2	cfail_get_composite_index.....	1029
2.5.4.3.3	cfail_compute_impact_	
	incidence_angle.....	1029
2.5.4.3.4	cfail_compute_side_hit	1030
2.5.4.3.5	normalize_x.....	1030
2.5.4.3.6	normalize_y.....	1031
2.5.4.3.7	compute_incidence_from_	
	back	1031
2.5.4.3.8	compute_incidence_from_	
	front.....	1032
2.5.4.3.9	compute_incidence_from_left...	1032
2.5.4.3.10	compute_incidence_from_	
	right	1033
2.5.4.4	c_ind_fire.c.....	1034
2.5.4.4.1	cfail_indirect_fire_damages	1035
2.5.4.4.2	cfail_get_indirect_index.....	1036
2.5.4.5	c_init.c	1037
2.5.4.5.1	cfail_init	1037
2.5.4.5.2	cfail_read_damage_file	1038
2.5.4.5.3	init_indirect_fire_table	1039
2.5.4.5.4	init_direct_fire_table	1040
2.5.4.5.5	cfail_damages_init	1041
2.5.4.6	cfail_loc.c	1042
2.5.4.7	cfail_loc.h.....	1042
2.5.4.8	f_break_sys.c.....	1043
2.5.4.8.1	fail_break_system.....	1044
2.5.4.8.2	fail_system_is_broken.....	1044
2.5.4.9	f_cat_kill.c.....	1045
2.5.4.9.1	fail_cat_kill.....	1045
2.5.4.9.2	fail_vehicle_is_destroyed.....	1046
2.5.4.10	f_dth_stat.c	1046
2.5.4.10.1	fail_death_status.....	1047
2.5.4.11	f_init.c	1047

	2.5.4.11.1	fail_table_init	1047
	2.5.4.11.2	fail_init_failure.....	1048
	2.5.4.11.3	fail_init	1049
2.5.4.12	f_reincarn.c.....		1050
	2.5.4.12.1	fail_reincarnation	1050
2.5.4.13	f_simul.c.....		1051
	2.5.4.13.1	fail_simul.....	1051
2.5.4.14	f_subsys.c		1052
	2.5.4.14.1	fail_subsys_init.....	1053
	2.5.4.14.2	fail_set_subsys	1053
	2.5.4.14.3	fail_failure_exists.....	1053
	2.5.4.14.4	fail_clear_subsys	1054
	2.5.4.14.5	fail_get_perm_subsys.....	1054
	2.5.4.14.6	fail_is_component_broken	1054
	2.5.4.14.7	fail_get_delta_subsystems.....	1055
	2.5.4.14.8	fail_set_subsys_bit	1055
	2.5.4.14.9	fail_clear_subsys_bit.....	1056
2.5.4.15	fail.h.....		1057
2.5.4.16	fail_loc.c		1057
2.5.4.17	fail_loc.h.....		1057
2.5.4.18	rand.c		1058
2.5.4.19	repair.c.....		1059
	2.5.4.19.1	lrepair_init	1059
	2.5.4.19.2	repair_uninit	1060
	2.5.4.19.3	repair_fix_system	1060
	2.5.4.19.4	repair_system_is_fixed.....	1061
	2.5.4.19.5	repair_fix_failure.....	1061
	2.5.4.19.6	repair_complete_system.....	1062
	2.5.4.19.7	repair_all_systems.....	1062
	2.5.4.19.8	repair_start_self_repair.....	1063
2.5.4.20	s_curr_cond.c		1063
	2.5.4.20.1	get_curr_condition	1064
2.5.4.21	s_event.c		1064
	2.5.4.21.1	sfail_event_occurred	1065
2.5.4.22	s_fixed.c		1066
	2.5.4.22.1	sfail_fixed_good_as_new.....	1066
2.5.4.23	s_init.c		1066
	2.5.4.23.1	sfail_init.....	1067
2.5.4.24	sfail_loc.c		1068
2.5.4.25	sfail_loc.h.....		1068
2.5.4.26	sfail_mnt_cond.c		1069
	2.5.4.26.1	sfail_maint_cond	1069
	2.5.4.26.2	sfail_maintenance_condition.....	1069

2.5.5	libturret.....	1070
2.5.5.1	libturret.h.....	1070
2.5.5.2	turret.c.....	1071
2.5.5.2.1	turret_stops_init.....	1072
2.5.5.2.2	turret_pos_init.....	1073
2.5.5.2.3	turret_set_stab_sys.....	1073
2.5.5.2.4	turret_set_stab_vector.....	1074
2.5.5.2.5	turret_get_stab_changes.....	1074
2.5.5.2.6	turret_move_azimuth.....	1075
2.5.5.2.7	turret_move_elevation.....	1076
2.5.5.2.8	turret_elevate_sight.....	1077
2.5.5.2.9	turret_elevate_gun.....	1077
2.5.5.2.10	elevate_system.....	1078
2.5.5.2.11	turret_sync_gun_with_sight.....	1079
2.5.5.2.12	turret_synch_sight_with_gun....	1080
2.5.5.2.13	turret_get_g_to_w.....	1081
2.5.5.2.14	turret_get_network_elevation....	1082
2.5.5.2.15	turret_get_network_azimuth.....	1082
2.5.5.2.16	turret_get_ref_ind.....	1083
2.5.5.2.17	turret_null_azimuth_ind.....	1083
2.5.5.2.18	turret_send_azimuth_ind.....	1083
2.5.5.2.19	turret_get_azimuth_str.....	1083
2.5.5.2.20	turret_update_check.....	1083
2.5.5.2.21	turret_update_rva.....	1084
2.5.5.2.22	turret_get_sight_in_world.....	1084
2.5.6	libsusp.....	1085
2.5.6.1	gun_fired.c.....	1085
2.5.6.1.1	suspension_gun_fired.....	1085
2.5.6.2	libsusp.h.....	1086
2.5.6.3	susp_accel.c.....	1086
2.5.6.3.1	suspension_acceleration_is.....	1086
2.5.6.4	susp_init.c.....	1087
2.5.6.4.1	suspension_uninit.....	1087
2.5.6.4.2	suspension_init.....	1087
2.5.6.5	susp_params.c.....	1088
2.5.6.5.1	suspension_params.....	1088
2.5.6.6	susp_simul.c.....	1090
2.5.6.6.1	suspension.....	1090
2.5.6.7	veh_init.c.....	1092
2.5.6.7.1	suspension_veh_init.....	1093
2.5.6.8	sus_loc.h.....	1094
2.5.7	libdyn.....	1095
2.5.7.1	calc_inert.c.....	1095

	2.5.7.1.1	dynamics_calc_inertial_forces	1095
2.5.7.2	calc_u.c		1096
	2.5.7.2.1	dynamics_calc_u	1096
2.5.7.3	calc_udot.c		1097
	2.5.7.3.1	dynamics_calc_udot	1097
2.5.7.4	filter.c		1098
	2.5.7.4.1	dynamics_filter_init	1098
	2.5.7.4.2	dynamics_filter_open	1099
	2.5.7.4.3	dynamics_filter_update	1100
2.5.7.5	init.c		1101
	2.5.7.5.1	dynamics_init	1101
	2.5.7.5.2	dump_mass	1101
2.5.7.6	lag.c		1102
	2.5.7.6.1	first_order_lag	1102
2.5.8	libkin		1103
2.5.8.1	hull_info.c		1103
	2.5.8.1.1	kinematics_get_w_to_h	1103
	2.5.8.1.2	kinematics_get_h_to_w	1104
	2.5.8.1.3	kinematics_get_h_to_o	1104
	2.5.8.1.4	kinematics_get_o_to_h	1105
	2.5.8.1.5	kinematics_get_u_norm	1105
	2.5.8.1.6	kinematics_get_velocity	1106
	2.5.8.1.7	kinematics_get_d_pos	1106
	2.5.8.1.8	kinematics_get_slope_ind	1107
2.5.8.3	kin_init.c		1108
	2.5.8.3.1	kinematics_uninit	1108
	2.5.8.3.2	kinematics_init	1109
2.5.8.4	kin_loc.c		1110
2.5.8.5	kin_loc.h		1110
2.5.8.6	kin_simul.c		1110
	2.5.8.6.1	kinematics_simul	1111
2.5.8.7	move_veh.c		1112
	2.5.8.7.1	kinematics_move_vehicle	1113
2.5.8.8	p_c_sines.c		1114
	2.5.8.5.1	kinematics_cant_cos	1114
	2.5.8.5.2	kinematics_pitch_cos	1115
	2.5.8.5.3	kinematics_cant_sin	1115
	2.5.8.5.1	kinematics_pitch_sin	1116
2.5.8.9	set_loc_kin.c		1116
	2.5.8.9.1	kinematics_set_local_kinematics	1117
	2.5.8.9.2	kinematics_fix_matrix	1118

	2.5.8.9.3	get_orient_vecs.....	1119
2.5.8.10	sqr_range.c		1119
	2.5.8.10.1	kinematics_range_squared	1120
2.5.8.11	turn_veh.c.....		1120
	2.5.8.11.1	kinematics_turn_vehicle.....	1121
2.5.8.12	update.c		1122
	2.5.8.12.1	kinematics_update_rva.....	1122
2.5.8.13	veh_init.c.....		1122
	2.5.8.13.1	kinematics_pos_init.....	1123
	2.5.8.13.2	kinematics_vehicle_init.....	1124
2.5.9	libhull		1125
	2.5.9.1 hull_init.c.....		1125
	2.5.9.1.1	hull_init	1125
	2.5.9.1.2	hull_uninit	1125
	2.5.9.2 hull_loc.c.....		1126
2.5.10	libbigwh.....		1127
	2.5.10.1 bigwh_init.c.....		1127
	2.5.10.1.1	bigwheel_uninit.....	1127
	2.5.10.1.2	bigwheel_init.....	1128
2.5.10.2	bigwh_loc.h.....		1128
2.5.10.3	calc_u_norm.c		1129
	2.5.10.3.1	bigwheel_calc_unit_normal	1129
2.5.10.4	chk_coll.c		1130
	2.5.10.4.1	collision_left_collision	1130
	2.5.10.4.2	collision_right_collision.....	1130
	2.5.10.4.3	collision_rear_collision	1131
2.5.10.5	coll_init.c.....		1131
	2.5.10.5.1	collision_init.....	1131
2.5.10.6	collision.c		1132
	2.5.10.6.1	collision_check_veh_coll_at	1133
	2.5.10.6.2	collision_cleared.....	1134
	2.5.10.6.3	collision_detected.....	1134
	2.5.10.6.4	collision_forget_about.....	1135
2.5.10.7	init_suppt.c.....		1136
	2.5.10.7.1	bigwheel_init_support_plane	1136
	2.5.10.7.2	bigwheel_init_height.....	1137
2.5.10.8	set_suppt.c		1137
	2.5.10.8.1	bigwheel_set_support_plane	1138
	2.5.10.8.2	reg_gnd_wheel	1139
	2.5.10.8.3	get_height_under_wheel	1140
2.5.10.9	sqr_range.c		1141
	2.5.10.9.1	compute_sqr_range	1141
2.5.10.10	tracks_stat.c		1142

	2.5.10.10.1	bigwheel_left_track_broken.....	1142
	2.5.10.10.2	bigwheel_right_track_broken....	1142
	2.5.10.10.3	bigwheel_repair_tracks	1142
2.5.10.11	veh_init.c		1142
2.5.11	libterrain		1143
	2.5.11.1	calc_elev.c	1144
	2.5.11.1.1	terrain_calc_elev	1145
	2.5.11.1.2	check_polys_incl	1146
	2.5.11.1.3	check_bvols_incl	1147
	2.5.11.1.4	terrain_get_height	1148
	2.5.11.1.5	terrain_inside	1149
	2.5.11.1.6	terrain_make_normal	1151
	2.5.11.1.7	terrain_make_edges	1151
2.5.11.2	get_size.c		1152
	2.5.11.2.1	terrain_get_patch_size	1152
2.5.11.3	get_soil.c		1153
	2.5.11.3.1	terrain_get_terrain_type	1153
2.5.11.4	lt_init.c		1153
	2.5.11.4.1	terrain_lt_init	1154
2.5.11.5	obstacles.c		1154
	2.5.11.5.1	terrain_obstructed	1155
2.5.11.6	preproc.c		1156
	2.5.11.6.1	terrain_preproc_terrain	1156
	2.5.11.6.2	terrain_add_poly_ptr	1158
	2.5.11.6.3	terrain_add_bvol_ptr	1159
2.5.11.7	terr_init.c		1159
2.5.11.8	terrain_loc.h		1159
2.5.11.9	verb_mode.c		1160
	2.5.11.9.1	terrain_verbose_mode_on	1160
2.5.12	librva		1161
	2.5.12.1	adj_veh_app.c	1161
	2.5.12.1.1	rva_adjust_veh_appear	1162
	2.5.12.1.2	rva_reset_veh_appear	1163
2.5.12.2	debug.c		1163
	2.5.12.2.1	rva_turn_debug_on	1163
	2.5.12.2.2	rva_turn_debug_off	1163
	2.5.12.2.3	rva_dump_priority_lists	1164
2.5.12.3	forget_veh.c		1164
	2.5.12.3.1	rva_forget_about_vehicle	1165
	2.5.12.3.2	delete_vehicles_from_list	1166
2.5.12.4	get_air_vehs.c		1167
	2.5.12.4.1	rva_get_air_veh_list	1167
2.5.12.5	get_obj_type.c		1168

	2.5.12.5.1	rva_get_object_type	1168
2.5.12.6	get_prior_list.c.....		1168
	2.5.12.6.1	rva_get_priority_list	1169
2.5.12.7	get_vap.c		1170
	2.5.12.7.1	rva_get_veh_app_pkt	1170
	2.5.12.7.2	rva_get_rva_entry.....	1171
2.5.12.8	get_veh_loc.c.....		1171
	2.5.12.8.1	rva_get_veh_loc	1171
2.5.12.9	get_vehs.c.....		1172
	2.5.12.9.1	rva_get_close_list.....	1172
	2.5.12.9.2	rva_get_lists	1173
	2.5.12.9.3	rva_get_num_hash_entries.....	1173
	2.5.12.9.4	rva_get_num_close_vehs	1173
	2.5.12.9.5	rva_get_num_air_vehs	1174
	2.5.12.9.6	set_save_num_static_vehs	1174
	2.5.12.9.7	rva_get_num_static_vehs	1174
	2.5.12.9.8	rva_get_num_mvg_vehs	1174
2.5.12.10	get_vid.c		1175
	2.5.12.10.1	rva_get_veh_id	1175
2.5.12.11	hash.c.....		1176
	2.5.12.11.1	rva_alloc_hash_table.....	1176
	2.5.12.11.2	rva_init_hash_table	1177
	2.5.12.11.3	rva_lookup_hash_table_entry ...	1177
	2.5.12.11.4	rva_remove_hash_table_entry ..	1178
	2.5.12.11.5	rva_insert_hash_table_entry.....	1179
	2.5.12.11.6	find_hash_value	1180
	2.5.12.11.7	free_hash_entry	1180
	2.5.12.11.8	get_hash_entry	1181
	2.5.12.11.9	rva_alloc_rva_table	1181
	2.5.12.11.10	rva_init_rva_table.....	1181
	2.5.12.11.11	rva_find_hash_entry.....	1182
	2.5.12.11.12	rva_delete_hash_entry.....	1182
	2.5.12.11.13	rva_add_hash_entry	1182
2.5.12.12	lock_veh.c		1183
	2.5.12.12.1	rva_lock_veh_into_buf.....	1183
	2.5.12.12.2	rva_unlock_veh	1184
2.5.12.13	markers.c		1185
	2.5.12.13.1	rva_alloc_marker_table.....	1185
	2.5.12.13.2	rva_init_marker_table	1185
	2.5.12.13.3	rva_process_markers.....	1186
	2.5.12.13.4	rva_process_unknown_ marker.....	1187
	2.5.12.13.5	adjust_markers	1187

2.5.12.14	prior_init.c	1188
2.5.12.14.1	rva_priority_setup	1189
2.5.12.15	prior_lists.c	1191
2.5.12.15.1	check_very_close_veh	1191
2.5.12.15.2	update_and_dead_reckon	1192
2.5.12.15.3	delete_or_timeout	1192
2.5.12.15.4	rotate_rwa_blades	1192
2.5.12.15.5	adjust_dynamic_vehicles	1194
2.5.12.15.6	adjust_static_vehicles	1195
2.5.12.16	prior_loc.c	1195
2.5.12.17	prior_loc.h	1196
2.5.12.18	prior_rm.c	1196
2.5.12.18.1	try_to_remove_veh	1197
2.5.12.19	prior_sort.c	1197
2.5.12.20	proc_update.c	1197
2.5.12.20.1	process_known_static	1198
2.5.12.20.2	process_unknown_static	1199
2.5.12.20.3	process_known_dynamic	1200
2.5.12.20.4	process_unknown_dynamic	1201
2.5.12.20.5	rva_process_update	1202
2.5.12.21	range_sqrd.c	1202
2.5.12.21.1	cig_get_current_range_sqrd	1202
2.5.12.22	rva_init.c	1203
2.5.12.22.1	rva_init	1203
2.5.12.23	rva_loc.c	1203
2.5.12.24	rva_loc.h	1204
2.5.12.25	rva_setup.c	1205
2.5.12.25.1	rva_setup	1205
2.5.12.26	show_vehs.c	1205
2.5.12.26.1	rva_vehicle_is_visible	1206
2.5.12.26.2	rva_vehicle_is_invisible	1207
2.5.12.26.3	vehicle_is_visible	1207
2.5.12.26.4	vehicle_is_invisible	1208
2.5.12.27	tell_cig.c	1208
2.5.12.27.1	rva_tell_cig_about_other_vehicles	1209
2.5.12.28	too_many_vehs.c	1209
2.5.12.28.1	cig_too_many_vehicles	1209
2.5.12.29	zero_veh.c	1209
2.5.12.29.1	zero_init_veh	1211
2.5.12.29.2	zero_uninit_veh	1211
2.5.12.29.3	zero_get_new_velocities	1212
2.5.12.29.4	zero_process_dynamic	1213

	2.5.12.29.5	zero_dead_reckon.....	1214	
	2.5.12.29.6	zero_set_extrapolation_ period.....	1214	
2.5.13	librva_util		1215	
	2.5.13.1	get_list.c	1215	
		2.5.13.1.1	rva_create_output_list.....	1215
		2.5.13.1.2	rva_get_output_list.....	1215
		2.5.13.1.3	rva_util_get_veh_app_pkt.....	1216
	2.5.13.2	librva_util.h		1217
	2.5.14	libfilter		1218
	2.5.14.1	add.c		1218
		2.5.15.1.1	filter_add_class.....	1218
	2.5.14.2	bounds.c.....		1219
		2.5.14.2.1	filter_change_class_bound	1219
	2.5.14.3	data.c		1219
	2.5.14.4	dump.c		1220
		2.5.14.4.1	filter_dump_filter_info.....	1220
	2.5.14.5	filter.c		1220
		2.5.14.5.1	do_packet_from_network.....	1221
		2.5.14.5.2	do_packet_from_host.....	1222
		2.5.14.5.3	do_init	1222
	2.5.14.6	force.c.....		1222
		2.5.14.6.1	filter_set_force	1223
	2.5.14.7	init.c.....		1223
		2.5.14.7.1	filter_init.....	1224
	2.5.14.8	location.c		1225
		2.5.14.8.1	filter_set_filter_threshold	1225
		2.5.14.8.2	filter_set_max_cig_range	1225
2.5.15	libimpacts			1226
	2.5.15.1	impacts.c.....		1226
		2.5.15.1.1	impacts_init.....	1226
		2.5.15.1.2	impacts_tell_cig_about_ impacts	1227
		2.5.15.1.3	impacts_queue_effect.....	1228
		2.5.15.1.4	impacts_get_element.....	1228
		2.5.15.1.5	impacts_free_element.....	1228
	2.5.15.2	libimps.h.....		1229
2.5.16	libapp			1230
	2.5.16.1	libapp.h.....		1230
	2.5.16.2	read.c		1231
		2.5.16.2.1	ReadDiscrepancyThresholds.....	1232
		2.5.16.2.2	ReadThreshold	1233
	2.5.16.3	thresh.c		1233

	2.5.16.3.1	PrepareDiscrepancyThreshold ..	1234
	2.5.16.3.2	AppearanceDiscrepancy ExceedsThresholds.....	1235
	2.5.16.3.3	clear_monitor_variables.....	1235
	2.5.16.3.4	get_reason_time	1236
	2.5.16.3.5	get_reason_app.....	1236
	2.5.16.3.6	get_reason_tur_azi	1236
	2.5.16.3.7	get_reason_gun_elev	1237
	2.5.16.3.8	get_reason_loc.....	1237
	2.5.16.3.5	get_reason_rot	1237
	2.5.16.3.6	print_reasons	1237
2.5.17	libnear.....		1238
	2.5.17.1	near_point.c.....	1238
		2.5.17.1.1	near_get_next_veh_near_ point.....
			1239
		2.5.17.1.2	near_get_veh_if_still_near_ point.....
			1240
		2.5.17.1.3	near_get_veh_closest_to_ point.....
			1241
		2.5.17.1.4	near_get_preferred_veh_ near_point.....
			1242
	2.5.17.2	near_vector.c	1242
		2.5.17.2.1	near_get_next_veh_near_ vector.....
			1243
		2.5.17.2.2	near_get_veh_if_still_near_ vector.....
			1244
		2.5.17.2.3	near_get_veh_closest_to_ vector.....
			1245
		2.5.17.2.4	near_get_preferred_veh_ near_vector
			1246
2.5.18	librotate.....		1247
	2.5.18.1	librot_loc.h	1247
	2.5.18.2	librotate.h.....	1248
	2.5.18.3	rot_comm.c.....	1250
		2.5.18.3.1	rotate_init_cig_element.....
			1251
		2.5.18.3.2	rotate_reassign_cig_element
			1252
		2.5.18.3.3	rotate_reset_cig_list
			1252
		2.5.18.3.4	rotate_get_cig_info.....
			1253
		2.5.18.3.5	rotate_send_msgs
			1254
		2.5.18.3.6	hull.....
			1254
		2.5.18.3.7	rotate_hull_init
			1255
		2.5.18.3.8	rotate_hull_simul.....
			1255
	2.5.18.4	rot_element.c	1256
		2.5.18.4.1	rotate_allocate_element.....
			1256
		2.5.18.4.2	rotate_init_element.....
			1257

2.5.18.4.3	rotate_init_stab_family.....	1259
2.5.18.4.4	rotate_init_stab_orphan.....	1260
2.5.18.4.5	rotate_init_stab_element.....	1261
2.5.18.4.6	rotate_init_offset_element.....	1261
2.5.18.4.7	rotate_prioritize_elements.....	1262
2.5.18.4.8	rotate_set_child_priority.....	1263
2.5.18.4.9	rotate_set_stops.....	1264
2.5.18.4.10	rotate_set_max_rate.....	1264
2.5.18.4.11	rotate_set_dynamic_ characteristics.....	1265
2.5.18.4.12	rotate_set_dynamic_state.....	1266
2.5.18.4.13	rotate_set_pre_command_ function.....	1267
2.5.18.4.14	rotate_set_post_command_ function.....	1268
2.5.18.4.15	rotate_set_no_rotate.....	1268
2.5.18.4.16	rotate_set_mat.....	1269
2.5.18.4.17	rotate_set_angle.....	1269
2.5.18.4.18	rotate_set_rate.....	1270
2.5.18.4.19	rotate_set_angle_and_rate.....	1270
2.5.18.4.20	rotate_set_current_angle.....	1271
2.5.18.4.21	rotate_modify_stab_offset.....	1271
2.5.18.4.22	rotate_set_stab_vector.....	1272
2.5.18.4.23	rotate_set_stab_vector_in_ coordinates.....	1272
2.5.18.4.24	rotate_set_stab_current_ position.....	1274
2.5.18.4.25	rotate_set_stab_current_ position_in_coordinates.....	1274
2.5.18.4.26	rotate_set_stab_point.....	1275
2.5.18.4.27	rotate_set_stab_point_ in_coordinates.....	1275
2.5.18.4.28	rotate_set_stab_rate.....	1277
2.5.18.4.29	rotate_set_stab_rate_in_ coordinates.....	1277
2.5.18.4.30	rotate_set_loc.....	1278
2.5.18.4.31	rotate_get_angle.....	1279
2.5.18.4.32	rotate_get_sin_angle.....	1279
2.5.18.4.33	rotate_get_cos_angle.....	1279
2.5.18.4.34	rotate_get_rate.....	1280
2.5.18.5	rot_relate.c.....	1281
2.5.18.5.1	rotate_relate_init.....	1283
2.5.18.5.2	rotate_number_node.....	1287
2.5.18.5.3	rotate_fill_permanent_tree.....	1288
2.5.18.5.4	rotate_relate_simul.....	1289

	2.5.18.5.5	rotate_get_mat.....	1290
	2.5.18.5.6	rotate_find_transform_path.....	1291
	2.5.18.5.7	rotate_path_val.....	1292
	2.5.18.5.8	rotate_save_path.....	1293
	2.5.18.5.9	rotate_mat.....	1293
	2.5.18.5.10	rotate_set_transform.....	1295
	2.5.18.5.11	rotate_transform_index	1296
	2.5.18.5.12	rotate_get_loc	1297
	2.5.18.5.13	rotate_set_location	1299
	2.5.18.5.14	rotate_location_index	1299
	2.5.18.5.15	rotate_break_links	1301
	2.5.18.5.16	dump_transform	1302
	2.5.18.5.17	dump_location.....	1302
	2.5.18.5.18	dump_break_list	1302
	2.5.18.5.19	relate_dump_transforms.....	1303
	2.5.18.5.20	relate_dump_locations	1303
2.5.18.6	rot_util.c		1304
	2.5.18.6.1	rotate_init	1304
	2.5.18.6.2	rotate_init_check	1304
	2.5.18.6.3	rotate_simul.....	1305
	2.5.18.6.4	rotate_exec	1306
	2.5.18.6.5	rotate_become_legal.....	1308
	2.5.18.6.6	rotate_stab	1309
	2.5.18.6.7	rotate_valid_angle	1310
	2.5.18.6.8	world.....	1310
2.5.19	libupdate		1311
	2.5.19.1	libupdate.c	1311
	2.5.19.1.1	vehicle_update.....	1312
	2.5.19.1.2	vehicle_place.....	1313
	2.5.19.1.3	vehicle_init.....	1314
	2.5.19.1.4	vehicle_set_position	1314
	2.5.19.1.4	vehicle_set_orientation.....	1315
	2.5.19.1.5	kinematics_set_orientation_ matrix	1315
	2.5.19.1.6	vehicle_mass_init.....	1316
	2.5.19.1.7	vehicle_restart	1316
	2.5.19.1.8	vehicle_A_acceleration	1316
	2.5.19.1.9	vehicle_B_acceleration	1317
	2.5.19.1.10	vehicle_A_velocity.....	1317
	2.5.19.1.11	vehicle_B_velocity.....	1317
	2.5.19.1.12	vehicle_velocity_magnitude.....	1318
	2.5.19.1.13	vehicle_A_r	1318
	2.5.19.1.14	vehicle_angular_velocity	1318

	2.5.19.1.15	vehicle_A_p	1318
	2.5.19.1.16	vehicle_B_s	1319
	2.5.19.1.17	vehicle_b2	1319
	2.5.19.1.18	vehicle_A_C_B	1319
	2.5.19.1.19	vehicle_B_C_A	1319
	2.5.19.1.20	vehicle_gravity_vector	1320
	2.5.19.1.21	vehicle_altitude	1320
	2.5.19.1.22	vehicle_climb_rate	1320
	2.5.19.1.23	vehicle_freeze	1320
	2.5.19.1.24	vehicle_thaw	1320
	2.5.19.1.25	vehicle_freeze_disable	1320
	2.5.19.1.26	vehicle_torques	1321
	2.5.19.1.27	vehicle_forces	1321
	2.5.19.1.28	freeze_state	1321
	2.5.19.1.29	dump routines	1322
	2.5.19.1.30	vehicle_set_init_state	1322
2.6	SIMULATION SUPPORT UTILITIES		1323
2.6.1	libmath		1325
	2.6.1.1	bivar_dist.c	1325
		2.6.1.1.1 bivariate_normal_distribution ...	1325
	2.6.1.2	cubic_func.c	1326
		2.6.1.2.1 find_cubic_func	1326
		2.6.1.2.2 cubic_func	1327
	2.6.1.3	inv_sin_cos.c	1327
		2.6.1.3.1 inv_sin_cos_deg	1328
		2.6.1.3.2 inv_sin_cos_rad	1328
	2.6.1.4	least_sq_fit.c	1329
		2.6.1.4.1 least_squares_fit	1330
		2.6.1.4.2 allocate_x_powers	1331
		2.6.1.4.3 allocate_y_powers	1332
		2.6.1.4.4 allocate_sim_lin_eq	1332
		2.6.1.4.5 generate_x_powers	1333
		2.6.1.4.6 generate_y_powers	1333
		2.6.1.4.7 generate_sim_lin_eq	1334
		2.6.1.4.8 solve_sim_lin_eq	1334
		2.6.1.4.9 generate_output_coeff_vals	1335
	2.6.1.5	libmath.h	1335
	2.6.1.6	limit.c	1336
		2.6.1.6.1 real_limit	1336
		2.6.1.6.2 int_limit	1336
	2.6.1.7	scaled_rand.c	1337
		2.6.1.7.1 scaled_rand	1337
2.6.2	libmatrix		1338

2.6.2.1	d2f_m_copy.c	1338
	2.6.2.1.1 d2f_mat_copy	1338
2.6.2.2	d2f_v_copy.c	1339
	2.6.2.2.1 d2f_vec_copy	1339
2.6.2.3	elr_copy.c	1339
	2.6.2.3.1 elr_copy	1339
2.6.2.4	elr_elr_cat.c	1340
	2.6.2.4.1 elr_elr_cat	1340
2.6.2.5	elr_form.c	1341
	2.6.2.5.1 elr_form	1341
2.6.2.6	elr_ident.c	1342
	2.6.2.6.1 elr_ident	1342
2.6.2.7	elr_to_mat.c	1343
	2.6.2.7.1 elr_to_mat	1343
2.6.2.8	elr_transp.c	1344
	2.6.2.8.1 elr_transpose	1344
2.6.2.9	f2d_m_copy	1344
	2.6.2.9.1 f2d_mat_copy	1344
2.6.2.10	f2d_v_copy.c	1345
	2.6.2.10.1 f2d_vec_copy	1345
2.6.2.11	fm_check.c	1346
	2.6.2.11.1 fmat_check	1346
2.6.2.12	fm_copy.c	1347
	2.6.2.12.1 fmat_copy	1347
2.6.2.13	fm_id_init.c	1348
	2.6.2.13.1 fmat_ident_init	1348
2.6.2.14	fm_m_mul.c	1349
	2.6.2.14.1 fmat_mat_mul	1349
2.6.2.15	fm_r_init.c	1350
	2.6.2.15.1 fmat_rot_init	1350
2.6.2.16	fmat_dump.c	1350
	2.6.2.16.1 fmat_dump	1350
2.6.2.17	fmat_r_init2.c	1351
	2.6.2.17.1 fmat_rot_init2	1351
2.6.2.18	fmat_sub.c	1352
	2.6.2.18.1 fmat_sub	1352
2.6.2.19	fmat_transp.c	1353
	2.6.2.19.1 fmat_transpose	1353
2.6.2.20	fv_check.c	1354
	2.6.2.20.1 fvec_check	1354
2.6.2.21	fv_d_prod.c	1355
	2.6.2.21.1 fvec_dot_prod	1355
2.6.2.22	fv_m_mul.c	1356

	2.6.2.22.1	fvec_mat_mul.....	1356
2.6.2.23		fv_scale.c.....	1357
	2.6.2.23.1	fvec_scale.....	1357
2.6.2.24		fv_x_prod.c	1358
	2.6.2.24.1	fvec_cross_prod	1358
2.6.2.25		fvec_add.c	1358
	2.6.2.25.1	fvec_add	1358
2.6.2.26		fvec_copy.c	1359
	2.6.2.26.1	fvec_copy	1359
2.6.2.27		fvec_dump.....	1359
	2.6.2.27.1	fvec_dump.....	1359
2.6.2.28		fvec_norm.c.....	1360
	2.6.2.28.1	fvec_normalize	1360
2.6.2.29		fvec_sub.c.....	1361
	2.6.2.29.1	fvec_sub	1361
2.6.2.30		m_fix_m.c	1362
	2.6.2.30.1	mat_fix_matrix	1362
2.6.2.31		m_id_init.c.....	1363
	2.6.2.31.1	mat_ident_init.....	1363
2.6.2.32		m_m_mul.c.....	1364
	2.6.2.32.1	mat_mat_mul.....	1364
2.6.2.33		m_r_int2.c	1365
	2.6.2.33.1	mat_rot_init2	1365
2.6.2.34		m_trig_init.c.....	1366
	2.6.2.34.1	mat_trig_init.....	1366
2.6.2.35		m_v_mul.c.....	1367
	2.6.2.35.1	mat_vec_mul	1367
2.6.2.36		mat_add.c	1368
	2.6.2.36.1	mat_add	1368
2.6.2.37		mat_adj.c	1369
	2.6.2.37.1	mat_adjugate	1369
2.6.2.38		mat_check.c.....	1370
	2.6.2.38.1	mat_check.....	1370
2.6.2.39		mat_copy.c	1371
	2.6.2.39.1	mat_copy	1371
2.6.2.40		mat_deter.c	1372
	2.6.2.40.1	mat_determinant.....	1372
2.6.2.41		mat_dump.c.....	1373
	2.6.2.41.1	mat_dump.....	1373
2.6.2.42		mat_form.c	1373
	2.6.2.42.1	mat_form	1373
2.6.2.43		mat_ident.c	1374
	2.6.2.43.1	mat_ident.....	1374

2.6.2.44	mat_init.c.....	1375
	2.6.2.44.1 mat_init.....	1375
2.6.2.45	mat_inv.c.....	1376
	2.6.2.45.1 mat_inverse.....	1376
2.6.2.46	mat_lev_init.c.....	1377
	2.6.2.46.1 mat_level_init.....	1377
2.6.2.47	mat_r_init.c.....	1378
	2.6.2.47.1 mat_rot_init.....	1378
2.6.2.48	mat_scale.c.....	1379
	2.6.2.48.1 mat_scale.....	1379
2.6.2.49	mat_sub.c.....	1380
	2.6.2.49.1 mat_sub.....	1380
2.6.2.50	mat_to_elr.c.....	1381
	2.6.2.50.1 mat_to_elr.....	1381
2.6.2.51	mat_transp.c.....	1382
	2.6.2.51.1 mat_transpose.....	1382
2.6.2.52	new_m_m_mul.c.....	1383
	2.6.2.52.1 nmat_mat_mul.....	1383
2.6.2.53	v_cos_prod.c.....	1384
	2.6.2.53.1 vec_cos_prod.....	1384
2.6.2.54	v_dot_prod.c.....	1385
	2.6.2.54.1 vec_dot_prod.....	1385
2.6.2.55	v_e_transf.c.....	1386
	2.6.2.55.1 vec_elr_transform.....	1386
2.6.2.56	v_m_mul.c.....	1387
	2.6.2.56.1 vec_mat_mul.....	1387
2.6.2.57	vec_add.c.....	1388
	2.6.2.57.1 vec_add.....	1388
2.6.2.58	vec_check.c.....	1389
	2.6.2.58.1 vec_check.....	1389
2.6.2.59	vec_copy.c.....	1390
	2.6.2.59.1 vec_copy.....	1390
2.6.2.60	vec_dump.c.....	1390
	2.6.2.60.1 vec_dump.....	1390
2.6.2.61	vec_init.c.....	1391
	2.6.2.61.1 vec_init.....	1391
2.6.2.62	vec_neg.c.....	1391
	2.6.2.62.1 vec_neg.....	1391
2.6.2.63	vec_norm.c.....	1392
	2.6.2.63.1 vec_normalize.....	1392
2.6.2.64	vec_scale.c.....	1393
	2.6.2.64.1 vec_scale.....	1393
2.6.2.65	vec_sub.c.....	1393

	2.6.2.65.1	vec_sub.....	1393
2.6.2.66		vec_x_prod.c	1394
	2.6.2.66.1	vec_cross_prod.....	1394
2.6.2.67		libmatrix.h	1394
2.6.3		libtimers.....	1395
2.6.3.1		t_cur_tick.c	1395
	2.6.3.1.1	timers_get_current_tick.....	1395
2.6.3.2		t_cur_time.c.....	1396
	2.6.3.2.1	timers_get_current_time.....	1396
2.6.3.3		t_data.c	1396
	2.6.3.3.1	timers_get_data	1396
2.6.3.4		t_del_proc.c	1397
	2.6.3.4.1	timers_delay_proc	1397
2.6.3.5		t_free.c.....	1398
	2.6.3.5.1	timers_free_timer	1398
2.6.3.6		t_get_timer.c.....	1399
	2.6.3.6.1	timers_get_timer.....	1399
2.6.3.7		t_in_use.c.....	1400
	2.6.3.7.1	timers_get_in_use_status	1400
2.6.3.8		t_init.c.....	1401
	2.6.3.8.1	timers_init.....	1401
2.6.3.9		t_loc.c	1401
2.6.3.10		t_milli.c	1402
	2.6.3.10.1	timers_elapsed_milliseconds.....	1402
2.6.3.11		t_null_proc.c.....	1403
	2.6.3.11.1	timers_null_proc.....	1403
2.6.3.12		t_reset.c	1403
	2.6.3.12.1	timers_reset_timeout_edge.....	1403
2.6.3.13		t_restart.c.....	1403
	2.6.3.13.1	timers_restart_timer	1404
2.6.3.14		t_set_null.c	1404
	2.6.3.14.1	timers_set_null_timer.....	1404
2.6.3.15		t_simul.c	1404
	2.6.3.15.1	timers_simul	1405
2.6.3.16		t_start.c	1406
	2.6.3.16.1	timers_init_starttime.....	1406
2.6.3.17		t_status.c.....	1407
	2.6.3.17.1	timers_status.....	1407
2.6.3.18		t_stop.c	1407
	2.6.3.18.1	timers_stop_timer.....	1407
2.6.3.19		t_stopped.c.....	1408
	2.6.3.19.1	timers_get_stopped_status	1408
2.6.3.20		t_ticking.c.....	1409

	2.6.3.20.1	timers_get_ticking_status	1409
2.6.3.21	t_ticks.c		1410
	2.6.3.21	timers_get_ticks_left.	1410
2.6.3.22	t_timeout.c		1411
	2.6.3.22.1	timers_get_timeout_edge	1411
2.6.3.23	timers_loc.h		1411
2.6.4	libutil		1412
2.6.4.1	beep.c		1412
	2.6.4.1.1	beep	1412
2.6.4.2	cp_2_TF1.c		1412
	2.6.4.2.1	copy_to_TF1	1412
2.6.4.3	cp_R4P3D.c		1413
	2.6.4.3.1	copy_R4P3D	1413
2.6.4.5	cp_TF1.c		1414
	2.6.4.5.1	copy_TF1	1414
2.6.4.6	cp_TF2.c		1414
	2.6.4.6.1	copy_TF2	1414
2.6.4.7	cp_Xrot2TF2.c		1415
	2.6.4.7.1	copy_X_rot_to_TF2	1415
2.6.4.8	cp_Yrot2TF2.c		1415
	2.6.4.8.1	copy_Y_rot_to_TF2	1415
2.6.4.9	cp_Zrot2TF2.c		1416
	2.6.4.9.1	copy_Z_rot_to_TF2	1416
2.6.4.10	database.c		1416
	2.6.4.10.1	util_set_database_name	1416
	2.6.4.10.2	util_get_database_name	1417
2.6.4.11	dead_zone.c		1417
	2.6.4.11.1	add_dead_zone	1417
2.6.4.12	deg.c		1418
	2.6.4.12.1	sin_cos_to_deg	1418
2.6.4.13	dump_core.c		1418
	2.6.4.13.1	dump_core	1419
2.6.4.14	error_printf.c		1419
	2.6.4.14.1	error_printf	1419
2.6.4.15	format.c		1419
	2.6.4.15.1	strchr	1420
	2.6.4.15.2	find_arg_type	1421
	2.6.4.15.3	format_decoder	1422
	2.6.4.15.4	copybuf	1423
2.6.4.16	libutil.h		1423
2.6.4.17	pr_R4P3D.c		1423
	2.6.4.17.1	print_R4P3D	1423
2.6.4.18	pr_TF1.c		1424

	2.6.4.18.1	print_TF1.....	1424
2.6.4.19	pr_TF2.c.....		1424
	2.6.4.19.1	print_TF2.....	1424
2.6.4.21	strtok.c.....		1425
	2.6.4.21.1	strtok_skip.....	1425
	2.6.4.21.2	strtok_find.....	1426
	2.6.4.21.3	strtok.....	1426
2.6.4.22	t_mat_dump.c.....		1427
	2.6.4.22.1	timed_mat_dump.....	1427
2.6.4.24	t_vec_dump.c.....		1427
	2.6.4.23.1	timed_vec_dump.....	1427
2.6.4.24	timed_printf.c.....		1428
	2.6.4.24.1	timed_printf.....	1428
	2.6.4.24.2	timed_printf_set.....	1428
2.6.5	libshm.....		1429
2.6.5.1	attach.c.....		1429
	2.6.5.1.1	attachshm.....	1430
2.6.5.2	detach.c.....		1431
	2.6.5.2.1	detachshm.....	1431
2.6.5.3	remove.c.....		1432
	2.6.5.3.1	removesshm.....	1432
2.6.5.4	shmcontrol.h.....		1432
2.6.6	libmove.....		1433
2.6.7	libser.....		1434
	2.6.7.1	ml_mem_dfn.h.....	1434
	2.6.7.2	ser_status.c.....	1434
	2.6.7.2.1	ser_heartbeat.....	1434
	2.6.7.2.2	ser_heartbeat_init.....	1434
2.6.8	libfifo.....		1435
2.6.8.1	f_dequeue.c.....		1435
	2.6.8.1.1	fifo_dequeue.....	1435
2.6.8.2	f_enqueue.c.....		1436
	2.6.8.2.1	fifo_enqueue.....	1436
2.6.8.3	f_init.c.....		1437
	2.6.8.3.1	fifo_init.....	1437
	2.6.8.3.2	fifo_uninit.....	1437
2.6.8.4	f_open_out.c.....		1437
	2.6.8.4.1	open_up_output_port.....	1437
	2.6.8.4.2	close_output_port.....	1438
2.6.8.5	f_print.c.....		1438
	2.6.8.5.1	fifo_print.....	1438
2.6.8.6	f_send_out.c.....		1439
	2.6.8.6.1	send_output_to_port.....	1439

2.6.8.7	fifo.h	1439
2.6.8.8	fifo_dfn.h	1440
2.6.9	libevent	1441
2.6.9.1	event.c	1441
	2.6.9.1.1 event_init_eventid	1441
	2.6.9.1.2 event_get_eventid	1441
2.6.9.2	libevent.h	1442
2.6.10	libveh	1443
	2.6.10.1 is_air_veh.c	1443
	2.6.10.1.1 is_air_vehicle	1443
	2.6.10.2 is_ammo_veh.c	1444
	2.6.10.2.1 is_ammo_vehicle	1444
	2.6.10.2.2 is_ammo_carrier	1445
	2.6.10.3 is_anti_air.c	1445
	2.6.10.3.1 is_anti_aircraft	1445
	2.6.10.4 is_apc.c	1446
	2.6.10.4.1 is_personnel_carrier	1446
	2.6.10.5 is_att_rwa.c	1447
	2.6.10.5.1 is_attack_rwa	1447
	2.6.10.6 is_friend.c	1448
	2.6.10.6.1 is_friendly	1448
	2.6.10.6.2 veh_set_force	1448
	2.6.10.6.3 veh_get_force	1448
	2.6.10.7 is_fuel_veh.c	1449
	2.6.10.7.1 is_fuel_vehicle	1449
	2.6.10.8 is_fwa.c	1450
	2.6.10.8.1 is_fwa	1450
	2.6.10.9 is_mb_tank.c	1451
	2.6.10.1.1 is_air_vehicle	1451
	2.6.10.10 is_rep_veh.c	1452
	2.6.10.10.1 is_repair_vehicle	1452
	2.6.10.11 is_rwa.c	1453
	2.6.10.11.1 is_rwa	1453
	2.6.10.12 libveh.h	1453
2.6.11	libmap	1454
2.6.11.1	damage.c	1454
	2.6.11.1.1 map_get_damage_files	1455
	2.6.11.1.2 check_for_nonexistant_	
	damage_files	1455
2.6.11.2	get_entry.c	1456
	2.6.11.2.1 map_get_ammo_entry_from_	
	network_type	1457
	2.6.11.2.2 search_obj_types	1458

2.6.11.2.3	map_get_network_type_ from_ammo_entry.....	1458
2.6.11.2.4	map_get_burst_ground_from_ ammo_entry.....	1459
2.6.11.2.5	map_get_burst_air_from_ ammo_entry.....	1459
2.6.11.2.6	map_get_burst_armor_from_ ammo_entry.....	1460
2.6.11.2.7	map_get_burst_wood_from_ ammo_entry.....	1460
2.6.11.2.8	map_get_burst_other_from_ ammo_entry.....	1461
2.6.11.2.9	map_get_tracer_from_ ammo_entry.....	1461
2.6.11.2.10	map_get_muzzle_flash_me_ from_ammo_entry.....	1462
2.6.11.2.11	map_get_muzzle_flash_other_ from_ammo_entry.....	1462
2.6.11.2.12	map_get_damage_file_index_ from_ammo_entry.....	1463
2.6.11.2.13	map_get_ammo_class_from_ ammo_entry.....	1463
2.6.11.2.14	map_is_bomb.....	1464
2.6.11.2.15	map_is_missile.....	1464
2.6.11.2.16	map_is_projectile.....	1465
2.6.11.3	map_ammo.c.....	1465
2.6.11.3.1	map_file_read.....	1466
2.6.11.3.2	read_entry_attributes.....	1466
2.6.11.3.3	skip_comment.....	1467
2.6.11.3.4	read_char.....	1467
2.6.11.3.5	read_long_int.....	1467
2.6.11.3.6	get_entries_until_end_ subclass.....	1468
2.6.11.3.7	check_for_defaults.....	1468
2.6.11.3.8	print_structure_contents.....	1468
2.6.11.4	map_asid.c.....	1469
2.6.11.4.1	map_read_asid_file.....	1469
2.6.11.4.2	map_set_asid.....	1470
2.6.11.4.3	map_clear_asid.....	1470
2.6.11.4.4	map_set_bumper_numbers.....	1470
2.6.11.4.5	map_set_dust_cloud.....	1471
2.6.11.4.6	map_get_bumper_status.....	1471
2.6.11.4.7	map_format_asid.....	1472
2.6.11.4.8	map_set_bumper_status.....	1472
2.6.11.5	map_vch.c ..	1473

	2.6.11.5.1	map_vehicle_file_read	1474
	2.6.11.5.2	read_vehicle_entry_attributes ...	1475
	2.6.11.5.3	skip_vch_comment.....	1475
	2.6.11.5.4	read_char_vehicle_entry	1475
	2.6.11.5.5	read_long_int_vehicle_entry	1476
	2.6.11.5.6	get_vehicle_entries_until_	
		end_subclass.....	1476
	2.6.11.5.7	check_for_vehicle_defaults.....	1476
	2.6.11.5.8	map_net_to_cig	1477
	2.6.11.5.9	check_for_match	1479
2.6.12	libmem.....		1480
	2.6.12.1	assign_mp.c	1480
	2.6.12.1.1	mem_assign_memory_ptr	1481
	2.6.12.1.2	mem_free_shared_memory	1481
	2.6.12.1.3	map_idc_values	1482
	2.6.12.1.4	unmap_idc_values	1482
	2.6.12.1.5	mem_get_idc_share_size	1482
	2.6.12.1.6	mem_get_memory_key	1482
	2.6.12.1.7	mem_get_total_share_size	1483
	2.6.12.2	assign sm.c	1483
	2.6.12.2.1	mem_assign_shared_memory ...	1483
2.6.13	m1_mem.c.....		1484
	2.6.13.1	mem_assign_other_ptrs.....	1484
2.6.14	m2_mem.c.....		1485
	2.6.14.1	mem_assign_other_ptrs.....	1485
2.6.15	kato_mem.c		1486
	2.6.15.1	mem_assign_other_ptrs.....	1486
2.6.16	librtc		1487
	2.6.16.1	rtc_timing.c	1487
	2.6.16.1.1	rtc_read_clock	1488
	2.6.16.1.2	rtc_start_time.....	1488
	2.6.16.1.3	rtc_stop_time.....	1489
	2.6.16.1.4	rtc_time_history	1489
	2.6.16.1.5	rtc_print_time	1490
	2.6.16.1.6	rtc_simul_history.....	1490
	2.6.16.1.7	rtc_print_overrun.....	1491
	2.6.16.1.8	rtc_printl	1491
	2.6.16.1.9	rtc_overrun	1491
	2.6.16.1.10	rtc_print_permanent	1491
	2.6.16.1.11	rtc_get_tick_rate.....	1492
	2.6.16.1.12	rtc_get_start.....	1492
	2.6.16.2	rtc.h.....	1492
2.6.17	libfile		1492

2.6.18	libquat.....	1493
2.6.18.1	calc_origin.c	1493
	2.6.18.1.1 kinematics_viewpoint_offset	1493
	2.6.18.1.2 kinematics_calc_origin_state	1494
2.6.18.2	calc_v.c.....	1495
	2.6.18.2.1 kinematics_calc_velocity	1495
2.6.18.3	form_c.x	1496
	2.6.18.3.1 kinematics_form_C.....	1496
2.6.18.4	form_N.c	1497
	2.6.18.4.1 kinematics_from_N.....	1497
2.6.18.5	form_e.c.....	1497
	2.6.18.5.1 kinematics_form_e	1497
2.6.18.6	form_g.c	1498
	2.6.18.6.1 kinematics_form_G.....	1498
2.6.18.7	form_r.c.....	1498
	2.6.18.7.1 kinematics_form_r	1498
2.6.18.8	form_s.c.....	1499
	2.6.18.8.1 kinematics_form_s	1499
2.6.18.9	make_e.c.....	1500
	2.6.18.9.1 make_e	1500
	2.6.18.9.2 quat_dump.....	1500
2.6.18.10	norm_e.c.....	1501
	2.6.18.10.1 normalize_e	1501
2.6.18.11	update_e.c.....	1501
	2.6.18.11.1 kinematics_update_e	1501
2.6.18.12	update_p.c	1502
	2.6.18.12.1 kinematics_update_p.....	1502
APPENDIX A: HEADER FILES		1503
SYSTEM LEVEL HEADER FILES.....		1503
VEHICLES HEADER FILES.....		1506
APPENDIX B: USER DEFINED TYPES.....		1511
APPENDIX C: MACROS		1523
APPENDIX D: GLOSSARY OF TERMS AND ABBREVIATIONS.....		1527
APPENDIX E: FUNCTIONS AND CALLING FUNCTIONS BY DIRECTORY AND FILE.....		E-1
INDEX BY SECTION NUMBER.....		Index-1

1 INTRODUCTION : VEHICLES CSCI

1.1 Background

→ The Vehicle Simulation software is the genesis of the distributed simulation concept. The first devices to interact over the SIMNET network were two M1 vehicle simulators. Gradually the concept diversified to add a greater variety of vehicles, as well as network analysis tools, command and control simulators, and Semi-Automated forces. ↘

A vehicle simulator is composed of one or more crewstations. Each crewstation provides the subset of the controls and indicators that would be available to that specific crew member on the actual vehicle. For example, on the M1 vehicle simulator the driver is provided with the steering bar, throttle, and brake controls necessary to drive the tank. A critical element of each crewstation is the out-the-window view, or vision block, associated with the station. These vision blocks provide the crew member with a three dimensional graphical representation of the terrain and objects in the environment external to the vehicle.

The purpose of the vehicle simulation software is to maintain the vehicle specific model, respond to crew inputs, update crewstation outputs, and communicate with the network. ←

The vehicle specific model includes the mathematical model of the drivetrain, powerplant, suspension, weapon systems, and any other internal systems necessary to compose the vehicle entity. These systems that are modeled have their corresponding operator interfaces in the crewstations. The crewstation outputs include the local sound system and the updates to the Computer Image Generator (CIG), which presents the three dimensional image in the vision blocks. Though it may be composed of several individual crewstations, the vehicle simulation represents a single entity on the network, and interacts with other network entities via the network interface.

In this document, the software for three vehicle simulations will be presented. These vehicles are:

- the M1 Abrams Tank
- the M2 Bradley Fighting Vehicle
- the Stealth (or Observation) Vehicle.

This document is intended to provide a "roadmap" through the M1 and M2 simulation and Stealth Vehicle code. It is divided into four sections. Section 1 provides a high level functional description of the Vehicles CSCI. It describes the interfaces of this CSCI with other CSCIs in the system, as well as the internal structure of the Vehicles CSCI.

Section 2 describes each of the first level CSCs which compose the Vehicles CSCI. Each first level CSC is broken down into its component CSCs and CSUs based on the functionality of the code. Each CSU is mapped to a library or a file. For each relevant procedure, information is provided about the parameters operated on, values returned, errors returned, callers of the procedure, and routines called by the procedure.

Section 2.1 contains information about the code that was generated to interface with external devices. Section 2.2 provides information about the major functions of the M1 simulation. Section 2.3 provides information about the major functions of the M2/M3 simulation. Section 2.4 provides information about the major functions of the Stealth Vehicle simulation. Section 2.5 provides information about the vehicle simulation libraries. Lastly, Section 2.6 discusses tools and utilities that are used throughout the simulation.

Section 3 describes the resources which are utilized by the Vehicles simulation, and Section 4 contains the source code listings.

1.2 External Interfaces

The vehicle simulation software is executed on a single processor. This processor is referred to as the Simulation Host. The host communicates via several external interfaces with other processors that compose a portion of the same vehicle simulator, or another simulation host. The number and type of interfaces depend on the vehicle being simulated and the architecture of the simulator.

The external interfaces for the M1, M2 and Stealth vehicle simulators are:

- the CIG
- the Simulation Network (SIMNET).

These interfaces are depicted in figures 1.2-1 and 1.2-2.

The simulator host tells the Computer Image Generator (CIG) what to display, that is, where on the terrain the simulated vehicle is located. It also tells the CIG the orientation of the simulated vehicle, the location and orientation of the other simulated vehicles, and where to display special effects such as bomb blasts and smoke. The CIG sends the simulator host descriptions of the local terrain around the simulated vehicle.

The simulator host describes the current vehicle state to the Management Command and Control (MCC), the other CSCIs, and the other simulators. They, in turn, keep it informed of the state of the entire exercise. These communications are carried out over SIMNET.

The communications schemes for the M1 and M2/M3 simulators and the Stealth Vehicle appear in Figures 1.2-1 and 1.2-2.

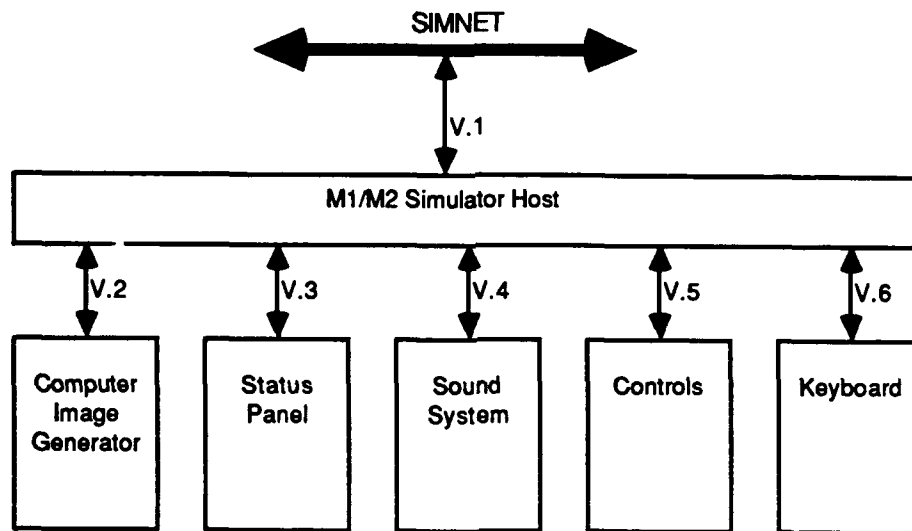


Figure 1.2-1 Communications Scheme for M1/M2 Simulators

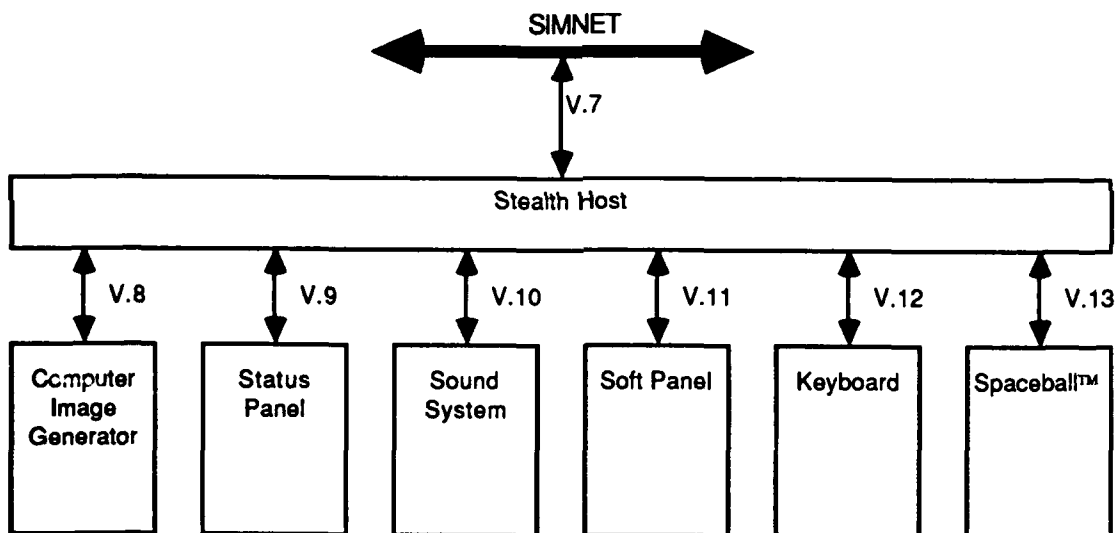


Figure 1.2-2: Communications Scheme for Stealth Simulation.

The vehicles communicate with other vehicles and other CSCIs over SIMNET using Protocol Data Units (PDUs). Each vehicle looks for (subscribes to) specific PDUs sent out by other vehicles and the other CSCIs. Each vehicle also sends out (broadcasts) PDUs of its own. Table 1.2-1 lists the PDUs which are of importance to the vehicles.

Table 1.2-1: PDU Information.

PDU Protocol Data Unit	CSCI Recieved From	CSCI Sent To	In Response To	Protocol Type	Transaction Type
Activate Request	MCC	Vehicle	Vehicle needs to be activated	Simulation	Transaction
Activate Response	Vehicle	MCC	Activate Request	Simulation	Transaction
Deactivate Request	Vehicle or Vehicle or MCC	MCC Vehicle Vehicle	Vehicle needs to be deactivated	Simulation	Transaction or Datagram
Deactivate Response	Vehicle	MCC	Deactivate Request	Simulation	Transaction
Vehicle Appearance	Vehicle	Multicast Group	Update or change vehicle appearance	Simulation	Datagram
Radiate	Vehicle	Multicast Group	Periodic	Simulation	Datagram
Fire	Vehicle	Multicast Group	Directed fire	Simulation	Datagram
Impact	Vehicle	Multicast Group	Impact of direct fire from firing vehicle	Simulation	Datagram or Transaction
Indirect Fire	MCC	Multicast Group	Non-direct fire	Simulation	Datagram
Collision	Colliding Vehicle or MCC	Colliding Vehicle Multicast Group	Vehicle collision	Simulation	Transaction or Datagram
Service Request	Vehicle	MCC	Need for munitions or repairs	Simulation	Datagram
Resupply Offer	MCC	Vehicle	Request for service	Simulation	Datagram
Resupply Received	Vehicle	MCC	Receipt of munitions	Simulation	Datagram
Resupply Cancel	Vehicle or MCC	MCC Vehicle	Vehicle no longer meets conditions for resupply	Simulation	Datagram
Repair Request	Vehicles	MCC	Repair process complete	Simulation	Transaction
Error	PVD	Stealth Vehicle	Any Stealth protocol command	Stealth	Transaction

Table 1.2-1: PDU Information (cont.).

PDU Protocol Data Unit	CSCI Received From	CSCI Sent To	In Response To	Protocol Type	Transaction Type
Visibility	PVD	Stealth Vehicle	Instructs Stealth Vehicle to change exercise ID's	Stealth	
Teleport	PVD	Stealth Vehicle	Tells Stealth Vehicle where to go	Stealth	
Attach	PVD	Stealth Vehicle	Tells what vehicle to attach to	Stealth	
Attached	Stealth Vehicle	PVD	After executing a successful attach operation	Stealth	
Mimic	PVD	Stealth Vehicle	To instruct the Stealth Vehicle to enter mimic mode	Stealth	
Appearance	Stealth Vehicle	PVD	Once every two seconds	Stealth	

The Vehicles communicate with the CIG through the CIG to Simulation Host Interface. This communication is accomplished through the use of data messages. Table 1.2-2 lists the data messages which are passed back and forth between the Vehicles and the CIG. (For additional information, reference *BBN Document #8912 - BBN GT100™ CIG to Simulation Host Interface Manual*.)

Table 1.2-2: CIG_SIM Message Information.

Data Message	CSCI Received From	CSCI Sent To	Message Type	Comment
CIG Control Message	Simulation Host	CIG	Control /Error Message	Instructs CIG to switch from one ready state to another
DR11 Packet Size Message	Simulation Host	CIG	Control /Error Message	At initialization or run time to change the size of incoming and outgoing DR11 packet buffers.
End Message	Simulation Host CIG	CIG Simulation Host	Control /Error Message	Indicates end of the contents in each packet buffer sent between the CIG and Simulation Host.
System Error Message	CIG	Simulation Host	Control /Error Message	In every packet buffer.
2-D Setup Message	Simulation Host	CIG	Configuration Message	Sets up subsystem's 2-D display data.
Create Configuration Node Message	Simulation Host	CIG	Configuration Message	Defines configuration node tree. Sent at initialization.
Define TX Mode Message	Simulation Host	CIG	Configuration Message	Used to set up viewport switching information for TX subsystems.
Overlay Setup Message	Simulation Host	CIG	Configuration Message	Sets up data required to display 3-D overlays for gun barrels.
Serial Device Close Message	Simulation Host	CIG	Configuration Message	Used during "prepare for simulation" state to close a channel to a subsystem.
Serial I/O Initialization Message	Simulation Host	CIG	Configuration Message	Used during "prepare for simulation" state to open a channel to a device for output during simulation state.
Trajectory Entry Transfer Message	Simulation Host	CIG	Configuration Message	Immediately follows Trajectory Table Transfer message; specifies each entry in trajectory table.
Trajectory Table Transfer Message	Simulation Host	CIG	Configuration Message	Used to load 30 Hz ballistics trajectory table into CIG for use during simulation.
Viewport State Message	Simulation Host	CIG	Configuration Message	Describes viewport attributes to CIG.
1 Rotation Message	Simulation Host	CIG	Simulation Support Message	Updates vehicle rotation.

Table 1.2-2: CIG_SIM Message Information (cont.).

Data Message	CSCI Received From	CSCI Sent To	Message Type	Comment
3 Rotations Message	Simulation Host	CIG	Simulation Support Message	Updates vehicle orientation.
Add Trajectory Table Message	Simulation Host	CIG	Simulation Support Message	Enhances the present Trajectory Table Transfer message.
AGL Setup Message	Simulation Host	CIG	Simulation Support Message	Triggers the CIG ballistics process into determining the height above ground and returns this value to the simulator host each frame.
Cancel Round Message	Simulation Host	CIG	Simulation Support Message	Allows simulator host to cancel a round that ballistics is processing.
Cloud State Message	Simulation Host	CIG	Simulation Support Message	Provides data needed to position the cloud models at the desired altitude.
Delete Trajectory Table Message	Simulation Host	CIG	Simulation Support Message	Allows simulation host to delete trajectory tables it downloaded earlier so that memory can be recovered and used for new trajectory tables.
Gun Overlay Message	Simulation Host	CIG	Simulation Support Message	Updates gun barrel position and gunner's information.
HPRXYZS Matrix Message	Simulation Host	CIG	Simulation Support Message	Sent each frame to update the "heading, Pitch, Roll, X, Y, Z, Scale" matrix associated with a configuration node.
Obscure Message	Simulation Host	CIG	Simulation Support Message	Displays a texture pattern on any viewport.
Other Vehicle State Message	Simulation Host	CIG	Simulation Support Message	Tracks each "other" vehicle within maximum viewing range of the simulator vehicle. Sent every frame for each vehicle until it moves beyond the maximum range.
Pass On Message	Simulation Host	CIG	Simulation Support Message	Provides the means to send data through the CIG system to a system embodied within the CIG.
Process Round Message	Simulation Host	CIG	Simulation Support Message	Enhances the Round Fired Message by sending additional data to the ballistics processor about a round.

Table 1.2-2: CIG_SIM Message Information (cont.).

Data Message	CSCI Received From	CSCI Sent To	Message Type	Comment
Request Laser Range Message	Simulation Host	CIG	Simulation Support Message	Sent when the simulation host requires laser range information for a viewport.
Request Point Information Message	Simulation Host	CIG	Simulation Support Message	Finds terrain characteristics at any point within the viewing range.
Rotate 2x1 Matrix Message	Simulation Host	CIG	Simulation Support Message	Updates vehicle position.
Round Fired Message	Simulation Host	CIG	Simulation Support Message	Sends data to a ballistics processor about a round.
RTS 4x3 Matrix Message	Simulation Host	CIG	Simulation Support Message	Updates vehicle position given the precalculated transformation matrix.
Scale Message	Simulation Host	CIG	Simulation Support Message	Not currently in use.
Serial I/O Write Message	Simulation Host	CIG	Simulation Support Message	Accepted during simulation state to write data to a device.
Show Effect Message	Simulation Host	CIG	Simulation Support Message	Commands the CIG to display an effect.
Static Vehicle Remove Message	Simulation Host	CIG	Simulation Support Message	Instructs CIG to remove a particular static vehicle from the display.
Static Vehicle State Message	Simulation Host	CIG	Simulation Support Message	Places static vehicles and obstacles in the simulation.
Subsystem Mode Message	Simulation Host	CIG	Simulation Support Message	Used by the simulation host to change subsystem specific parameters.
Terrain Feedback Initialization Header Message	Simulation Host	CIG	Simulation Support Message	Used to control the way the CIG builds the terrain feedback message.
Terrain Feedback Initialization Point Message	Simulation Host	CIG	Simulation Support Message	Sent after every Terrain Feedback Initialization Header Message. Specifies the points in vehicle coordinates from which the terrain feedback information is to be acquired.

Table 1.2-2: CIG_SIM Message Information (cont.).

Data Message	CSCI Received From	CSCI Sent To	Message Type	Comment
Terrain Feedback State Message	Simulation Host	CIG	Simulation Support Message	Tells the CIG to start and stop gathering terrain feedback information for a specific vehicle, or to change the frequency of processing for one's own vehicle.
Terrain Feedback Vehicle Position Message	Simulation Host	CIG	Simulation Support Message	Informs the CIG of new position and rotation of a simulated vehicle, other than one's own vehicle, for which terrain feedback data is being collected.
Trajectory Chord Message	Simulation Host	CIG	Simulation Support Message	Sends specific chords to the ballistics processor.
Trajectory Entry Message	Simulation Host	CIG	Simulation Support Message	Enhances the Trajectory Entry Transfer message. Sent after an Add Trajectory Table Message is sent.
Translation Message	Simulation Host	CIG	Simulation Support Message	Sent every frame to update the vehicle x, y, z position.
View Flags Message	Simulation Host	CIG	Simulation Support Message	Switches viewport paths on or off and changes branch values associated with the branch nodes.
View Magnification Message	Simulation Host	CIG	Simulation Support Message	Changes the field-of-view values specified in the Viewport State Message.
Viewport Update Message	Simulation Host	CIG	Simulation Support Message	Used to change CIG viewport-specific parameters.
AGL Message	CIG	Simulation Host	Simulation Support Message	Returns the height above ground of one's own vehicle centroid.
Hit Return Message	CIG	Simulation Host	Simulation Support Message	Returns the ballistics intersection information if a terrain polygon is struck.
Laser Return Message	CIG	Simulation Host	Simulation Support Message	Returns laser range information for the specified pixel within the screen each frame.
Local Terrain Message	CIG	Simulation Host	Simulation Support Message	Contains information which the simulator host uses to determine the position and orientation of the simulated vehicle and to calculate the dynamics of that vehicle.

Table 1.2-2: CIG_SIM Message Information (cont.).

Data Message	CSCI Received From	CSCI Sent To	Message Type	Comment
Local Terrain Place Message	CIG	Simulation Host	Simulation Support Message	Divides the Local Terrain message into predefined pieces that are transmitted one piece per frame.
Miss Message	CIG	Simulation Host	Simulation Support Message	This message is returned when a round specified by a Process Round Message or Round Fired Message, or a chord specified by a Trajectory Chord Message, does not intersect with any database element.
Pass Back Message	CIG	Simulation Host	Simulation Support Message	Returns information from an embedded processor to the simulation host. Applies only to TX subsystems.
Return Point Information Message	CIG	Simulation Host	Simulation Support Message	Returns information about requested points.
Terrain Feedback Header Message	CIG	Simulation Host	Simulation Support Message	Sent as the first message of a set of messages containing terrain feedback information for a specific vehicle.
Terrain Feedback Point Message	CIG	Simulation Host	Simulation Support Message	Sent as part of a set of messages containing terrain feedback information.
File Description Message	CIG or Simulation Host	Simulation Host CIG	System Support Message	Used to identify files or databases that will be transferred.
File Status Message	CIG or Simulation Host	Simulation Host CIG	System Support Message	Used to acknowledge that a block has been received, or to request re-transmission of the block.
File Transfer Message	CIG or Simulation Host	Simulation Host CIG	System Support Message	Used to transfer files and databases.
Test Name Message	Simulation Host	CIG	System Support Message	Requests that a test be run.

1.3 Internal Structure

The vehicle simulation software is a real time component that is executed at a fixed rate. Each pass through the real time loop is referred to as a frame. The frame event, the event that indicates the beginning of the frame, is controlled by the interface to the CIG. During a frame the vehicle simulation receives inputs from the external interfaces, updates the vehicle model accordingly, and updates the outputs to the external interfaces.

In addition, the internal interfaces are checked and updated. The crewstation controls are read and updated via the Interactive Device Controller (IDC). The sound system is also updated to complete the tactile queues for the crew members. During the frame, keyboard inputs are processed, primarily for debugging purposes, and outputs are generated to the simulator console if appropriate. The status panel indicators are altered if a change in the maintenance state of the simulator occurs.

A vehicle simulation is divided into six states:

- Start-Up
- Idle
- Simulation Initialize
- Simulation
- Simulation Stop
- Exit

During the Start-Up state, the mechanizations that are required to be performed only once when the application is run are performed. For example, communication devices are opened to the external interfaces, and blocks of memory are allocated at this time. The simulation transitions automatically from Start-Up to Idle state, where it waits for activation and the transition to Simulation Initialize state.

At Simulation Initialize state, those activities that must be performed once for each restart of the vehicle are performed. For example, the controls and indicators are placed in a known condition here.

In the Simulation state, the controls are active, and the vehicle models as well as all of the external interfaces are updated. From this state the simulation may transition to Simulation Stop, which transitions to Idle state, or Exit, which leaves the application altogether.

The organization of the Vehicles CSCI is shown in Figure 1.3-1.

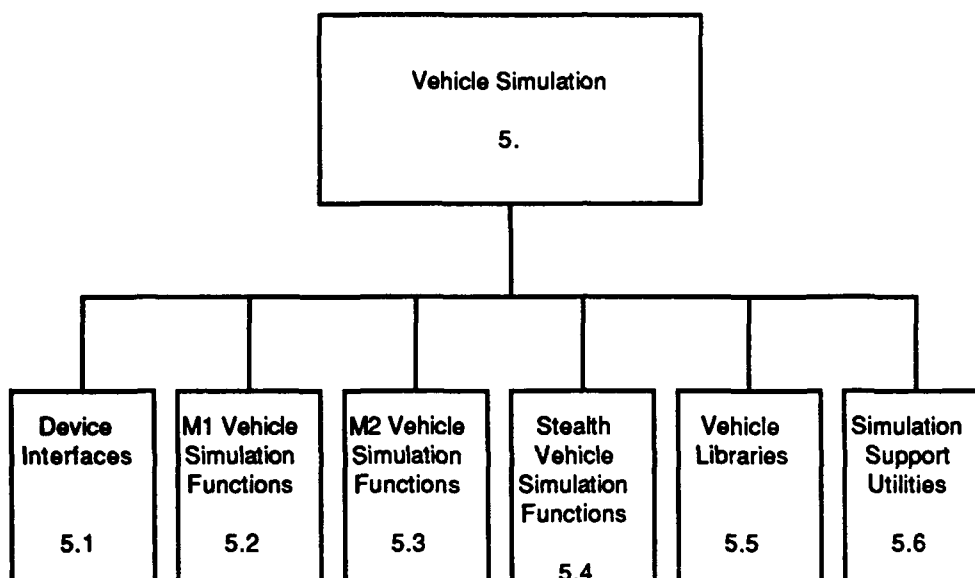


Figure 1.3-1 Structure of the Vehicles CSCI.

The following six (6) first level CSC's are associated with this CSCI:

- Device Interface Software
- M1 Vehicle Simulation Software
- M2 Device Simulation Software
- Stealth Vehicle Simulation Software
- Vehicle Libraries
- Simulation Support Utilities

1.4 Configuration and Configuration Management

The software components are grouped in libraries according to either the corresponding external interface, or the subsystem that is being simulated. These libraries are assembled into a hierarchy, with the most generic components (those shared by multiple vehicles) at the top, and the most vehicle specific components at the bottom. This software is written in the C Programming Language and is ported to several platforms.

The M1 simulator program host is an MC 5600 MASSCOMP™ computer based on the 68020 microprocessor. It sends command messages to and receives information from other parts of the simulator and system.

The M2 simulator program host is a BBN™ Butterfly GP1000™, based on the Motorola 68020 microprocessor. It sends command messages to and receives information from other parts of the simulator and system. Although the M1 and M2 simulator hosts are different machines, they provide the same functionality.

Two versions of the Stealth Vehicle are currently supported, Stealth 3 and Stealth 3GT. The Stealth 3 uses a BBN 120T CIG and MASSCOMP 5600 simulator host borrowed from an M1 simulator. The Stealth 3GT uses a BBN GT101™ CIG and MVME147 board host computer for image generation and host simulation.

1.5 Terminology and Documentation

The following documents provide additional information about the vehicle simulation and hardware:

BBN Report No. 6323 - SIMNET M1 Abrams Main Battle Tank Simulation; Software Description and Documentation (Revision 1)

BBN Report No. 6892 - SIMNET M2/3 Bradley Fighting Vehicle Simulation; Software Description and Documentation

BBN Report No. 7102 - The SIMNET Network and Protocols

BBN Report No. 7348 - SIMNET Stealth Vehicle Functional Specification and Operator's Manual

Top-level Descriptions for SIMNET Software, May 1990

The SIMNET Maintenance Manual

BBN Document No. 8912 - BBN GT100tm CIG to Simulation Host Interface Manual

A glossary of terms and abbreviations appears in Appendix D.

1.6 Miscellaneous

There are header files associated with each of the libraries and vehicle specific simulation files. These files are important for understanding the simulation code. They contain external references to routines that are externally accessible from the libraries. They contain external references to global variables and contain definitions of data types and constants for use as parameters. These files are listed in Appendix A.

2 CSC DESCRIPTIONS

2.1 Device Interface Software

Figure 2.1-1 illustrates the structure of the Device Interface Software CSC.

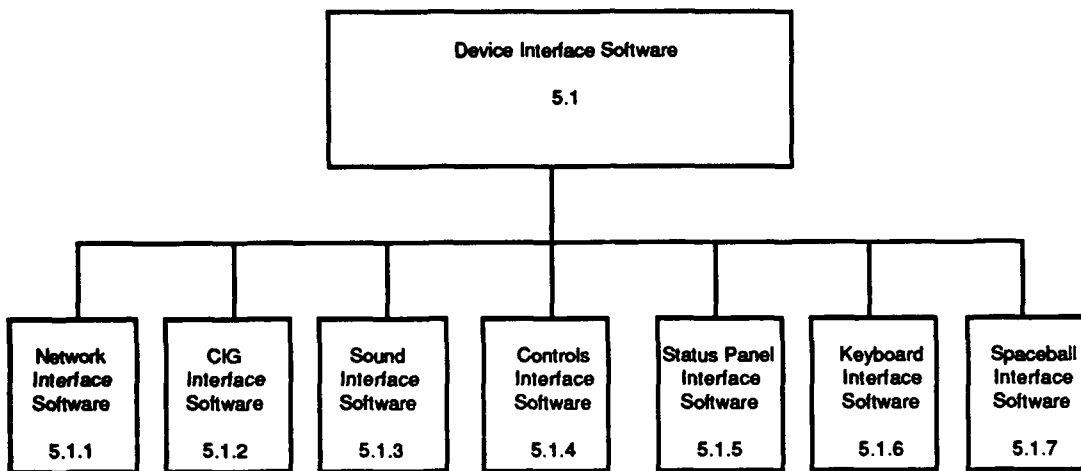


Figure 2.1-1: Device Interface Software structure.

The simulation host must communicate with a number of different devices. These devices include the Computer Image Generator (CIG), the SIMNET local area network (LAN), the sound system, the status panel, the controls, and the host keyboard (for development purposes). The term controls refers to the collection of controls, lights, switches, meters, etc. The Stealth Vehicle host must communicate with many of the above devices as well as with the Spaceball™.

Software modules provide interfaces to each of these devices. Since the M1, M2, and Stealth vehicle simulations need to communicate with this set of devices, the interfaces incorporate a significant amount of functionality required for vehicle simulation in general, allowing large amounts of software to be shared by all three applications. The individual modules are described below.

The second level CSCs associated with this CSC are as follows:

- Network Interface Software
- CIG Interface Software
- Sound Interface Software
- Controls Interface Software
- Status Panel Interface Software
- Keyboard Interface Software
- Spaceball™ Interface Software

2.1.1 Network Interface Software

Figure 2.1-2 shows the structure of the Network Interface Software CSC.

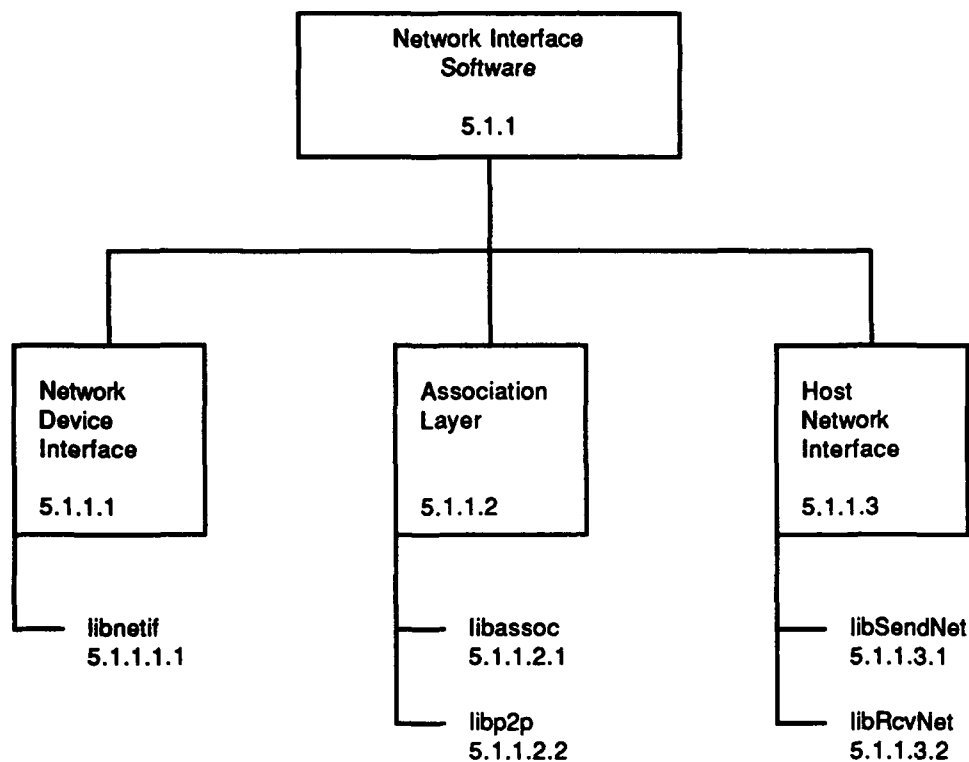


Figure 2.1-2: Network Interface Software.

The SIMNET LAN is accessed through a set of software modules organized as C archive libraries. These libraries provide a high level interface to the SIMNET network. For example, a function call can be made by the simulation host to inform the outside world of its current status. All of the internals of formatting the packet and moving it onto the network are handled by the network module.

This second level CSC is made up of the following third level CSCs:

Network Device Interface
Association Layer
Host Network Interface

Note that the "Called By" sections of the tables describing the routines contained in these CSCs will list only the routines that call them internal to the module.

2.1.1.1 Network Device Interface

A facility was required for sending and receiving data messages over SIMNET, initializing and stopping the network interface, testing the status of the interface, and manipulating shared memory buffers. This functionality is realized by the CSU libnetif.

2.1.1.1.1 libnetif

(./simnet/release/src/libsrc/libnetif [libnetif])

Libnetif provides an operating system and hardware platform independent interface to the network. The library interface is layered on top of the facilities provided by the operating system and hardware platform.

Libnetif provides facilities for sending and receiving datagrams, obtaining and zeroing network statistics, obtaining the local station address, manipulating network addresses, and specifying multicast addresses for receipt. Additionally, a set of control functions is available for initializing and stopping the network interface, testing the status of the interface, manipulating memory buffers shared between the application and the interface, and timestamping packets.

Reference Section 2.20.21 in the MCC CSCI SDD.

2.1.1.2 Association Layer

The SIMNET Association Layer provides network services to send/receive datagrams, and exactly once (xo) transactions. It also provides a subscription service to multicast addresses on the network, allowing an individual application to selectively accept/reject network traffic from different SIMNET exercises, and different protocols within an exercise. The CSUs that make up this CSC are libassoc and libp2p.

2.1.1.2.1 libassoc

(./simnet/release/src/libsrc/libassoc [libassoc])

Libassoc communicates with the network through libnetif to provide these services.

Reference Section 2.20.1 in the MCC CSCI SDD.

2.1.1.2.2 libp2p (./simnet/release/src/libsrc/libp2p [libp2p])

Point to point communication is accomplished by creating point to point datagrams. Point to point datagrams consist of a Simulation or Data Collection PDU with a point to point header. This library contains the routines used to initialize, send, and receive the point to point datagrams.

2.1.1.2.2.1 init.c (./simnet/release/src/libsrc/libp2p/init.c)

Includes:

"assoc.h"
"address.h"
"p_num.h"

Variable Declarations:

p2pSimAddress

2.1.1.2.2.1.1 PointToPointOpen

This routine contains the definitions needed to initialize the point to point layer.

Parameters		
Parameter	Type	Where Typedef Declared
device	pointer to char	Standard
def	pointer to char	Standard
Internal Variables		
Variable	Type	Where Typedef Declared
netHandle	int	Standard
Return Values		
Return Value	Type	Meaning
netHandle	int	procedure was successful; it will be used as a handle to all subsequent p2p or assoc calls
-1	int	procedure failed
Calls		
Function	Where Described	
AssocOpen	See MCC CSCI Section 2.20.1.5.1	
AssocSubscribe	See MCC CSCI Section 2.20.1.1.1	
AssocGetSimAddress	See MCC CSCI Section 2.20.1.9.1	

Table 2.1-1: PointToPointOpen Information.

2.1.1.2.2.2 p2p.h (./simnet/release/src/libsrc/libp2p/p2p.h)

This file contains all definitions needed by external modules to use the point to point layer library.

Procedure Declarations:

- PointToPointOpen
- PointToPointReceivePDU
- PointToPointSendPDU

2.1.1.2.2.3 p2p_local.c (./simnet/release/src/libsrc/libp2p/p2p_local.c)

This file contains all definitions needed locally to use the point to point layer library.

Defines:

- DATACOPY

External Declarations:

- p2pSimAddress

2.1.1.2.2.4 receive.c (./simnet/release/src/libsrc/libp2p/receive.c)

Includes:

```
"assoc.h"
"p_p2p.h"
"p_num.h"
"p2p_local.h"
```

2.1.1.2.2.4.1 PointToPointReceivePDU

This routine contains all definitions needed to receive packets from the SIMNET network through the point to point layer library. When a point to point datagram is received, its point to point header is removed, leaving the underlying simulation PDU.

Parameters		
Parameter	Type	Where Typedef Declared
handle	int	Standard
data	pointer to pointer to char	Standard
length	pointer to long int	Standard
group	pointer to MulticastGroupID	p_assoc.h
protocol	pointer to AssociationUserProtocol	p_assoc.h
primitive	pointer to long int	Standard
originator	pointer to SimulationAddress	address.h
transID	pointer to TransactionIdentifier	address.h
respondent	pointer to SimulationAddress	address.h
Internal Variables		
Variable	Type	Where Typedef Declared
p	pointer to PointToPointPDU	p_p2p.h
assocRetVal	int	Standard
Return Values		
Return Value	Type	Meaning
assocRetVal	int	The return value from the call to AssocReceive; if equal to 0: packet received if equal to 1: no packet
0	int	the packet was received
1	int	no packet was received
Calls		
Function	Where Described	
AssocReceivePDU	See MCC CSCI Section 2.20.1.6.1	
ASSOC ADDRESS EQUAL	assoc.h	
AssocCurrentlySubscribed	See MCC CSCI Section 2.20.1.1.3	

Table 2.1-2: PointToPointReceivePDU Information.

2.1.1.2.2.5 send.c

(./simnet/release/src/libsrc/libp2p/send.c)

Includes:

```
"network.h"
"p_assoc.h"
"p_p2p.h"
"p_num.h"
"p_size.h"
"assoc.h"
"p2p_local.h"
```

Declarations:

```
buf[(MAX_DATA_SIZE_8023 / sizeof(AssociationDataUnit)) + 1]
```

2.1.1.2.2.5.1 PointToPointSendPDU

This routine contains all definitions needed for sending point to point datagrams. Simulation PDUs are given a point to point header and sent through the point to point communications layer.

Parameters		
Parameter	Type	Where Typedef Declared
handle	int	Standard
data	pointer to char	Standard
length	long int	Standard
group	MulticastGroupID	p_assoc.h
protocol	AssociationUserProtocol	p_assoc.h
destination	pointer to SimulationAddress	address.h
Internal Variables		
Variable	Type	Where Typedef Declared
p	pointer to PointToPointPDU	p_p2p2.h
Return Values		
Return Value	Type	Meaning
0	int	procedure was successful
-1	int	procedure failed
Calls		
Function	Where Described	
DATACOPY	p2p_local.h (macro definition)	
AssocSendDatagram	See MCC CSCI SDD Section 2.20.1.2.1	
PRO_P2P_PDU_SIZE	p_size.h (macro definition)	

Table 2.1-3: PointToPointSendPDU Information.

2.1.1.3 Host Network Interface

Two archive libraries, libSendNet and libRcvNet, provide the host interface to libassoc. libSendNet provides routines for sending various types of SIMNET packets, while libRcvNet provides routines for receiving them.

When the simulation host is ready to read a packet from the network, it calls the function **process_a_packet()**, which can be found in "proc_a_pkt.c". **process_a_packet()** requests a packet from the Association Layer. If the request is successful, the function decodes the packet, determining its protocol and type, and then calls the appropriate function to process that type of packet. These packet-specific functions make up the remainder of libRcvNet.

The simulation host sends a packet onto the network by calling a packet-specific routine in libSendNet, and providing it with whatever information is necessary. These routines then use the Association Layer services to send the packet as either a datagram or transaction, as appropriate. The functionality of this CSC is realized by the following CSUs:

libSendNet
libRcvNet

2.1.1.3.1 libSendNet

(./simnet/release/src/vehicle/libsrc/libSendNet [libSendNet])

LibSendNet provides routines for sending various types of SIMNET packets.

2.1.1.3.1.1 act_rsp.c

(./simnet/release/src/vehicle/libsrc/libSendNet/act_rsp.c)

Includes:

"send_loc.h"
"pro_sim.h"
"pro_size.h"
"pro_num.h"
"pro_assoc.h"
"net_stats.h"
"veh_app_loc.h"

2.1.1.3.1.1.1 send_activate_response

This routine sends an activate transaction response PDU as an acknowledgement of receipt of an activate PDU. Parameters are represented as follows:

originator -- the simulation address of the originator
tid -- the transaction identifier

Parameters		
Parameter	Type	Where Typedef Declared
originator	pointer to SimulationAddress	address.h
tid	TransactionIdentifier	address.h
Internal Variables		
Variable	Type	Where Typedef Declared
response	SimulationPDU	p_sim.h
r	register pointer to ActivateResponseVariant	p_sim.h
Calis		
Function	Where Described	
veh_spec_activate_time		
fill_simHdr	Section 2.1.1.3.1.42.1	
AssocSendResponse	See MCC CSCI Section 2.20.1.4.2	
network_get_net_handle	Section 2.1.1.3.2.12.1	
PRO_SIM_ACTIVATE_RSP_SIZE	p_size.h (macro definition)	
NOTE SENT		

Table 2.1-4: send_activate_response Information.

2.1.1.3.1.2 activ_params.c

(./simnet/release/src/vehicle/libsrc/libSendNet/activ_params.c)

This file contains a stub for vehicles that do not use any of the vehicle specific activation parameters.

2.1.1.3.1.3 activate.c

(./simnet/release/src/vehicle/libsrc/libSendNet/activate.c)

This file is not implemented in the Version 6.6 release.

2.1.1.3.1.4 appearance.c

(./simnet/release/src/vehicle/libsrc/libSendNet/appearance.c)

The libSendNet library keeps track of a formatted vehicle appearance packet (VAP), or Vehicle Appearance PDU, for a simulated vehicle. Once the VAP is initially set up, large portions remain constant from tick to tick (i.e., exerciseID, vehicleID, etc.). Only certain fields will change during a frame (i.e., appearance, position, etc.). This file formats the dynamic parts of the vehicle appearance packet.

Includes:

- "send_loc.h"
- "sim_style.h"
- "pro_sim.h"
- "pro_stlth.h"
- "veh_type.h"
- "net_network.h"
- "libutil.h"
- "libkin.h"
- "libhull.h"
- "libmatrix.h"

Declarations:

- veh_should_be_scaled
- scale_mat[3][3]
- format_app_debug

2.1.1.3.1.4.1 format_vehicle_appearance

This routine is called from `net_xmit()` to format a vehicle appearance packet. All of the fields which remain constant during an exercise (exerciseID, vehicleID, alignment, etc.) should have been set up in `network_use_activation()` so that this routine only needs to fill in dynamic information like appearance and position. Note that it calls `fill_vehicle_spec_appearance()` to fill in vehicle-specific information.

Parameters		
Parameter	Type	Where Typedef Declared
pkt	pointer to SimulationPDU	p_sim.h
Internal Variables		
Variable	Type	Where Typedef Declared
vap	register pointer to VehicleAppearanceVariant	p_sim.h
Calls		
Function	Where Described	
d2f mat copy	Section 2.6.2.1.1	
kinematics get h to w	Section 2.5.8.2.2	
vec copy	Section 2.6.2.59.1	
kinematics get o to h	Section 2.5.8.2.4	
filter set location	Section 2.5.14	
d2f vec copy	Section 2.6.2.2.1	
kinematics get velocity	Section 2.5.8.2.6	
engine get speed	Section 2.2.6.2.2.11	
net current time	See MCC CSCI SDD Section 2.20.2.8.3	
network get net handle	Section 2.1.1.3.2.12.1	
fill_vehicle_spec_appearance	Section 2.2.7.1 Section 2.3.7.1 Section 2.4.6.1	

Table 2.1-5: format_vehicle_appearance Information.

2.1.1.3.1.4.2 format_stealth_appearance

This routine formats the dynamic portions of the stealth appearance PDU, in the same manner as the routine `format_vehicle_appearance()`.

Parameters		
Parameter	Type	Where Typedef Declared
pkt	pointer to StealthPDU	p_stlth.h
Internal Variables		
Variable	Type	Where Typedef Declared
sap	register pointer to StealthAppearanceVariant	p_stlth.h
id	pointer to VehicleID	basic.h
Calls		
Function	Where Described	
d2f mat copy	Section 2.6.2.1.1	
kinematics get h to w	Section 2.5.8.2.2	
network get vehicle id	Section 2.1.1.3.1.22.1	
veh get force	Section 2.6.10.6.3	
vec copy	Section 2.3.2.59.1	
kinematics get o to h	Section 2.5.8.2.4	
filter set location	Section 2.5.14	
d2f vec copy	Section 2.6.2.2.1	
kinematics get velocity	Section 2.5.8.2.6	
net current time	See MCC CSCI Section 2.20.2.8.3	
network get net handle	Section 2.1.1.3.2.12.1	
fill_vehicle_spec_appearance	Section 2.2.7.1 Section 2.3.7.1 Section 2.4.6.1	

Table 2.1-6: format_stealth_appearance Information.

2.1.1.3.1.5 citv_instr.c

(./simnet/release/src/vehicle/libsrc/libSendNet/citv_instr.c)

Code in this file is not compiled in the Version 6.6 release.

2.1.1.3.1.6 citv_orient.c

(./simnet/release/src/vehicle/libsrc/libSendNet/citv_orient.c)

Code in this file is not compiled in the Version 6.6 release.

2.1.1.3.1.7 coll_rsp.c

(./simnet/release/src/vehicle/libsrc/libSendNet/coll_rsp.c)

Includes:

"send_loc.h"
 "pro_sim.h"
 "pro_size.h"
 "pro_num.h"
 "pro_assoc.h"
 "veh_app_loc.h"

2.1.1.3.1.7.1 network_send_collision_response

This routine sends a transaction response to a Collision PDU.

Parameters		
Parameter	Type	Where Typedef Declared
originator	pointer to SimulationAddress	address.h
tid	TransactionIdentifier	address.h
Internal Variables		
Variable	Type	Where Typedef Declared
pdu	SimulationPDU	p_sim.h
Calls		
Function	Where Described	
network_fill_hdr_send_sim_rsp	Section 2.1.1.3.1.46.1	

Table 2.1-7: network_send_collision_response Information.

2.1.1.3.1.8 collision.c

(./simnet/release/src/vehicle/libsrc/libSendNet/collision.c)

Includes:

"send_loc.h"
 "pro_sim.h"
 "pro_size.h"
 "veh_app_loc.h"

2.1.1.3.1.8.1 network_send_outta_my_way_mf

This routine sends a Collision PDU as a transaction request when the invoking process detects a collision between the vehicle and another vehicle.

Parameters		
Parameter	Type	Where Typedef Declared
eventID	EventID	basic.h
the guy i hit	pointer to VehicleID	basic.h
Internal Variables		
Variable	Type	Where Typedef Declared
pdu	SimulationPDU	p_sim.h
pkt	register pointer to CollisionVariant	p_sim.h
Calls		
Function	Where Described	
network fill hdr send sim trans	Section 2.1.1.3.1.47.1	
PRO_SIM_COLLISION_SIZE	p_size.h	

Table 2.1-8: network_send_outta_my_way_mf Information.

2.1.1.3.1.9 deact_rsp.c

(./simnet/release/src/vehicle/libsrc/libSendNet/deact_rsp.c)

Includes:

"send_loc.h"
 "pro_sim.h"
 "pro_size.h"
 "pro_num.h"
 "pro_assoc.h"
 "veh_app_loc.h"

2.1.1.3.1.9.1 network_send_deactivate_response

This routine sends a transaction response to a Deactivate PDU.

Parameters		
Parameter	Type	Where Typedef Declared
originator	pointer to SimulationAddress	address.h
tid	TransactionIdentifier	address.h
Internal Variables		
Variable	Type	Where Typedef Declared
response	SimulationPDU	p_sim.h
r	register pointer to DeactivateResponseVariant	p_sim.h
Calls		
Function	Where Described	
network fill hdr send sim rsp	Section 2.1.1.3.1.46.1	
PRO SIM DEACTIVATE RSP SIZE	p_size.h	

Table 2.1-9: network_send_deactivate_response Information.

2.1.1.3.1.10 deactivate.c

(./simnet/release/src/vehicle/libsrc/libSendNet/deactivate.c)

Includes:

"send_loc.h"
 "pro_sim.h"
 "pro_size.h"
 "veh_app_loc.h"

2.1.1.3.1.10.1 send_deactivate_pkt

This routine sends a Deactivate Request PDU through the datagram service, in order to withdraw a vehicle from the exercise. Other vehicles receiving the Deactivate Request PDU will cease to dead reckon and display the vehicle.

Parameters		
Parameter	Type	Where Typedef Declared
reason	DeactivateReason	p_sim.h
Internal Variables		
Variable	Type	Where Typedef Declared
pdu	SimulationPDU	p_sim.h
pkt	pointer to DeactivateRequestVariant	p_sim.h
Calls		
Function	Where Described	
network fill_hdr send sim_pkt	Section 2.1.1.3.1.42.5	
PRO SIM DEACTIVATE REQ SIZE	p_size.h	

Table 2.1-10: send_deactivate_pkt Information.

2.1.1.3.1.11 death_status.c

(./simnet/release/src/vehicle/libsrc/libSendNet/death_status.c)

The values of the Vehicle Appearance PDU's appearance bit-field describe various effects to the vehicle's basic appearance. This file contains routines to set these bits before sending of the Vehicle Appearance PDU.

Includes:

"send_loc.h"
"veh_appear.h"

2.1.1.3.1.11.1 network_set_death_status

This routine sets the VAP appearance bit-field describing the vehicle's death status (i.e., whether the vehicle is destroyed).

Parameters		
Parameter	Type	Where Typedef Declared
we are dead	int	Standard

Table 2.1-11: network_set_death_status Information.

2.1.1.3.1.11.2 network_set_smoking_status

This routine sets the VAP appearance bit-field describing the vehicle's smoking status (i.e., whether a plume of smoke is rising from the vehicle).

Parameters		
Parameter	Type	Where Typedef Declared
new smoke	int	Standard

Table 2.1-12: network_set_smoking_status Information.

2.1.1.3.1.11.3 network_set_burning_status

This routine sets the VAP appearance bit-field describing the vehicle's burning status (i.e., whether flames are coming from the vehicle).

Parameters		
Parameter	Type	Where Typedef Declared
new burn	int	Standard

Table 2.1-13: network_set_burning_status Information.

2.1.1.3.1.11.4 network_set_commo_kill

This routine sets the VAP appearance bit-field describing whether the vehicle's communications appear to be disabled.

Parameters		
Parameter	Type	Where Typedef Declared
kill status	int	Standard

Table 2.1-14: network_set_commo_kill Information.

2.1.1.3.1.11.5 network_set_mobility_kill

This routine sets the VAP appearance bit-field describing whether the vehicle's mobility appears to be disabled.

Parameters		
Parameter	Type	Where Typedef Declared
kill status	int	Standard

Table 2.1-15: network_set_mobility_kill Information.

2.1.1.3.1.11.6 network_set_firepower_kill

This routine sets the VAP appearance bit-field describing whether the vehicle's firepower capability appears to be disabled.

Parameters		
Parameter	Type	Where Typedef Declared
kill status	int	Standard

Table 2.1-16: network_set_firepower_kill Information.

2.1.1.3.1.12 dust_status.c

(./simnet/release/src/vehicle/libsrc/libSendNet/dust_status.c)

Includes:

"send_loc.h"
"veh_appear.h"

2.1.1.3.1.12.1 network_set_dust_cloud

This routine sets a ground vehicle's VAP appearance bit-field indicating the appearance of a trailing dust cloud, and describing its size.

Parameters		
Parameter	Type	Where Typedef Declared
new_cloud	int	Standard

Table 2.1-17: network_set_dust_cloud Information.

2.1.1.3.1.13 event_flag.c

(./simnet/release/src/vehicle/libsrc/libSendNet/event_flag.c)

Includes:

"stdio.h"
"varargs.h"
"send_loc.h"
"veh_app_loc.h"
"pro_assoc.h"
"pro_data.h"
"pro_size.h"

2.1.1.3.1.13.1 network_send_event_flag

This routine sends an Event Flag PDU. The event flag (or event ID) is used by the Data Collection system to match up actions and events. For example, the event of a gun firing involves performing several actions, such as making sounds and showing effects (i.e., muzzle flash, impact, and burning and smoking of an impacted vehicle). This routine uses the Standard C VARARG capability for handling variable argument lists.

Parameters		
Parameter	Type	Where Typedef Declared
va_alist	va_dcl	Standard (variable argument list declaration)
Internal Variables		
Variable	Type	Where Typedef Declared
flags	int	Standard
fmt	pointer to char	Standard
ap	va list	
buf[maxNetworkPDUSize / sizeof(long)]	long	Standard
pdu	pointer to DataCollectionPDU	p_data.h
pkt	register pointer to EventFlagVariant	p_data.h
text	pointer to char	Standard
apdu	pointer to AssociationPDU	p_assoc.h
max_size	int	Standard
buf_size	int	Standard
malloc()	extern pointer to char	Standard
Calls		
Function	Where Described	
PRO ASSOC DATAGRAM HDR SIZE	p_size.h	
PRO DATA EVENT FLAG SIZE	p_size.h	
format_decoder	Section 2.6.4.15.3	
network_fill_hdr_send_dc_pkt	Section 2.1.1.3.1.42.6	
PRO DATA EVENT FLAG SIZE	p_size.h	

Table 2.1-18: network_send_event_flag Information.

2.1.1.3.1.14 ex_status.c

(./simnet/release/src/vehicle/libsrc/libSendNet/ex_status.c)

Includes:

```
"sys/types.h"
"sys/timeb.h"
"send_loc.h"
"veh_app_loc.h"
"pro_data.h"
"pro_assoc.h"
"pro_size.h"
"pro_num.h"
"net_stats.h"
```

2.1.1.3.1.14.1 send_exercise_status_pkt

This routine sends an Exercise Status PDU using the datagram service. Normally, this routine will be used to provide information for restarting a simulator after an equipment failure or for restoring an exercise to earlier conditions.

Parameters		
Parameter	Type	Where Typedef Declared
exercise	ExerciseID	basic.h
Internal Variables		
Variable	Type	Where Typedef Declared
pdu	DataCollectionPDU	p_data.h
pkt	register pointer to ExerciseStatusVariant	p_data.h
curr_time	struct timeb	
Calls		
Function	Where Described	
timers elapsed milliseconds	Section 2.6.3.10.1	
AssocSendDatagram	See MCC CSCI Section 2.20.1.2.1	
network_get_net_handle	Section 2.1.1.3.2.12.1	
PRO DATA EXERCISE STATUS SIZE	p_size.h	
NOTE SENT		

Table 2.1-19: send_exercise_status_pkt Information.

2.1.1.3.1.14.2 send_exercise_status_trans

This routine sends an Exercise Status PDU using the transaction service. Normally, this routine will be used to provide information in response to a query.

Parameters		
Parameter	Type	Where Typedef Declared
originator	pointer to SimulationAddress	address.h
tid	TransactionIdentifier	address.h
exercise	ExerciseID	basic.h
Internal Variables		
Variable	Type	Where Typedef Declared
pdu	DataCollectionPDU	p_data.h
pkt	register pointer to ExerciseStatusVariant	p_data.h
current_time	struct timeb	
Calls		
Function	Where Described	
timers_elapsed milliseconds	Section 2.6.3.10.1	
AssocSendTransact	See MCC CSCI Section 2.20.1.4.1	
network_get_net_handle	Section 2.1.1.3.2.12.1	
PRO DATA EXERCISE STATUS SIZE	p_size.h	
NOTE_SENT		

Table 2.1-20: send_exercise_status_trans Information.

2.1.1.3.1.15 fuState.c

(./simnet/release/src/vehicle/libsrc/libSendNet/fuState.c)

Code in this file is not compiled in the Version 6.6 release.

2.1.1.3.1.16 get_exer_id.c

(./simnet/release/src/vehicle/libsrc/libSendNet/get_exer_id.c)

Includes:

"send_loc.h"

"veh_app_loc.h"

2.1.1.3.1.16.1 network_get_exercise_id

This routine returns the exercise ID field from the Vehicle Appearance PDU which is maintained by libSendNet. This routine is generally used as a utility by other modules in the vehicle code that need to access information in the VAP.

Return Values		
Return Value	Type	Meaning
app_pkt->exercise	ExerciseID	the exercise ID field of the VAP

Table 2.1-21: network_get_exercise_id Information.

2.1.1.3.1.17 get_force.c

(./simnet/release/src/vehicle/libsrc/libSendNet/get_force.c)

Includes:

"send_loc.h"

"veh_app_loc.h"

2.1.1.3.1.17.1 network_get_vehicle_force

This routine returns the vehicle force ID field from the Vehicle Appearance PDU which is maintained by libSendNet. This routine is generally used as a utility by other modules in the vehicle code that need to access information in the VAP.

Return Values		
Return Value	Type	Meaning
app_pkt->variant.appearance.force	ForceID	the force ID field of the VAP

Table 2.1-22: network_get_vehicle_force Information.

2.1.1.3.1.18 get_guises.c

(./simnet/release/src/vehicle/libsrc/libSendNet/get_guises.c)

Includes:

"pro_sim.h"

"send_loc.h"

"veh_app_loc.h"

2.1.1.3.1.18.1 network_get_vehicle_guises

This routine returns the vehicle guises field from the Vehicle Appearance PDU which is maintained by libSendNet. This routine is generally used as a utility by other modules in the vehicle code that need to access information in the VAP.

Return Values		
Return Value	Type	Meaning
&app_pkt->variant.appearance.guises	pointer to VehicleGuises	vehicle guise field

Table 2.1-23: network_get_vehicle_guises Information.

2.1.1.3.1.19 get_sim_type.c

(./simnet/release/src/vehicle/libsrc/libSendNet/get_sim_type.c)

Includes:

"send_loc.h"

"veh_app_loc.h"

2.1.1.3.1.19.1 network_get_simulator_type

This routine returns the simulation type field (M1 simulator, M2 simulator, etc.) from the Vehicle Appearance PDU which is maintained by libSendNet. This routine is generally used as a utility by other modules in the vehicle code that need to access information in the VAP.

Return Values		
Return Value	Type	Meaning
my_simulator_type	SimulatorType	type of my simulator.

Table 2.1-24: network_get_simulator_type Information.

2.1.1.3.1.20 get_unit.c

(./simnet/release/src/vehicle/libsrc/libSendNet/get_unit.c)

Includes:

"send_loc.h"

"veh_app_loc.h"

2.1.1.3.1.20.1 network_get_vehicle_unit

This routine returns the vehicle's organizational unit field from the Vehicle Appearance PDU which is maintained by libSendNet. This routine is generally used as a utility by other modules in the vehicle code that need to access information in the VAP.

Return Values		
Return Value	Type	Meaning
&init_pkt->unit	pointer to OrganizationalUnit	the unit field of the VAP

Table 2.1-25: network_get_vehicle_unit Information.

2.1.1.3.1.21 get_veh_app.c

(./simnet/release/src/vehicle/libsrc/libSendNet/get_veh_app.c)

Includes:

"pro_sim.h"
 "send_loc.h"
 "veh_app_loc.h"

2.1.1.3.1.21.1 network_get_vehicle_appearance

This routine returns the vehicle appearance field from the Vehicle Appearance PDU which is maintained by libSendNet. This routine is generally used as a utility by other modules in the vehicle code that need to access information in the VAP.

Return Values		
Return Value	Type	Meaning
app_pkt->variant.appearance.appearance	unsigned long	the appearance field of the VAP

Table 2.1-26: network_get_vehicle_appearance Information.

2.1.1.3.1.22 get_veh_id.c

(./simnet/release/src/vehicle/libsrc/libSendNet/get_veh_id.c)

Includes:

"send_loc.h"
 "veh_app_loc.h"

2.1.1.3.1.22.1 network_get_vehicle_id

This routine returns the vehicle ID field from the Vehicle Appearance PDU which is maintained by libSendNet. This routine is generally used as a utility by other modules in the vehicle code that need to access information in the VAP.

Return Values		
Return Value	Type	Meaning
&app_pkt->variant.appearance.vehicleID	pointer to VehicleID	basic.h

Table 2.1-27: network_get_vehicle_id Information.

2.1.1.3.1.23 get_veh_type.c

(./simnet/release/src/vehicle/libsrc/libSendNet/get_veh_type.c)

Includes:

"send_loc.h"

"veh_app_loc.h"

2.1.1.3.1.23.1 network_get_vehicle_type

This routine returns the vehicle type field from the Vehicle Appearance PDU which is maintained by libSendNet. This routine is generally used as a utility by other modules in the vehicle code that need to access information in the VAP.

Return Values		
Return Value	Type	Meaning
init_pkt->status.vehicleType	ObjectType	vehicle type field

Table 2.1-28: network_get_vehicle_type Information.

2.1.1.3.1.24 get_xmt_fail.c

(./simnet/release/src/vehicle/libsrc/libSendNet/get_xmt_fail.c)

Includes:

"send_loc.h"

2.1.1.3.1.24.1 net_xmt_failed

This routine returns the value of the *netxmt_failed* flag. It is used to determine whether regularly transmitted packets (i.e., appearance, equipment status, and simulation status packets) are actually being sent.

Return Values		
Return Value	Type	Meaning
netxmt_failed	BOOLEAN	if TRUE, the network transmit failed; if FALSE, the network transmit was successful

Table 2.1-29: net_xmt_failed Information.

2.1.1.3.1.25 gnd_impact.c

(./simnet/release/src/vehicle/libsrc/libSendNet/gnd_impact.c)

Includes:

```
"send_loc.h"
"pro_sim.h"
"pro_size.h"
"mun_type.h"
"sim_ammo.h"
"veh_app_loc.h"
"libmap.h"
"basic.h"
```

Declarations:

```
null_vehicleID
null_vehicle_coords
```

2.1.1.3.1.25.1 network_send_ground_impact

This routine sends an Impact PDU for a ground impact.

Parameters		
Parameter	Type	Where Typedef Declared
eventID	EventID	basic.h
ammo_type	int	Standard
detonator_type	ObjectType	p_sim.h
quantity	unsigned short	Standard
rate	unsigned short	Standard
location	pointer to WorldCoordinates	basic.h
range	double	Standard
Internal Variables		
Variable	Type	Where Typedef Declared
pdu	SimulationPDU	p_sim.h
pkt	register pointer to ImpactVariant	p_sim.h
Calls		
Function	Where Described	
map_get_network_type_from_amm_entry	Section 2.6.11.2.3	
vec_copy	Section 2.6.2.59.1	
fvec_copy	Section 2.6.2.26.1	
network_fill_hdr_send_sim_pkt	Section 2.1.1.3.1.42.5	
PRO SIM IMPACT SIZE	p_size.h	

Table 2.1-30: network_send_ground_impact Information.

2.1.1.3.1.26 imp_rsp.c

(./simnet/release/src/vehicle/libsrc/libSendNet/imp_rsp.c)

Includes:

```
"send_loc.h"
"pro_sim.h"
"pro_size.h"
"pro_num.h"
"pro_assoc.h"
"veh_app_loc.h"
```

2.1.1.3.1.26.1 network_send_impact_response

This routine sends a transaction response to an Impact PDU.

Parameters		
Parameter	Type	Where Typedef Declared
originator	pointer to SimulationAddress	address.h
tid	TransactionIdentifier	address.h
Internal Variables		
Variable	Type	Where Typedef Declared
pdu	SimulationPDU	p_sim.h
Calls		
Function	Where Described	
network_fill_hdr_send_sim_rsp	Section 2.1.1.3.1.46.1	

Table 2.1-31: network_send_impact_response Information.

2.1.1.3.1.27 laser_detect.c

(./simnet/release/src/vehicle/libsrc/libSendNet/laser_detect.c)

Code in this file is not compiled in the Version 6.6 release.

2.1.1.3.1.28 laser_fire.c

(./simnet/release/src/vehicle/libsrc/libSendNet/laser_fire.c)

Code in this file is not compiled in the Version 6.6 release.

2.1.1.3.1.29 laser_range.c

(./simnet/release/src/vehicle/libsrc/libSendNet/laser_range.c)

Includes:

"send_loc.h"
 "veh_app_loc.h"
 "pro_data.h"
 "pro_size.h"

2.1.1.3.1.29.1 network_send_laser_range

This routine sends a Laser Range PDU.

Parameters		
Parameter	Type	Where Typedef Declared
result	LaserRangeResult	p_data.h
LRswitch	ReturnSwitch	p_data.h
targetID	pointer to TargetDescriptor	basic.h
laser tip[]	double	Standard
location[]	double	Standard
Internal Variables		
Variable	Type	Where Typedef Declared
pdu	DataCollectionPDU	p_data.h
pkt	register pointer to LaserRangeVariant	p_data.h
Calls		
Function	Where Described	
network fill_hdr_send_dc_pkt	Section 2.1.1.3.1.42.6	
PRO_DATA_LASER_RANGE_SIZE	p_size.h	

Table 2.1-32: network_send_laser_range Information.

2.1.1.3.1.30 laser_result.c

(./simnet/release/src/vehicle/libsrc/libSendNet/laser_result.c)

Code in this file is not compiled in the Version 6.6 release.

2.1.1.3.1.31 missile.c

(.simnet/release/src/vehicle/libsrc/libSendNet/missile.c)

Includes:

```

"stdio.h"
"math.h"
"send_loc.h"
"veh_app_loc.h"
"pro_sim.h"
"pro_size.h"
"mun_type.h"
"sim_macros.h"
"mass_std.h"
"dgi_stdg.h"
"sim_cig_if.h"
"libmiss_dfn.h"
"timers_dfn.h"
"sim_ammo.h"

```

Defines:

```

MAX_DEAD_RECK_ERROR    -- in meters
FIRST_MISSILE_ID
LAST_MISSILE_ID

```

Typedefs:

```

MISSILE_INFO    -- contains missile information, including the eventID of the
                  missile in flight, the dead reckoned position of the missile,
                  the velocity, and the first chord for the missile.

```

Declarations:

```

in_flight[MAX_MISSILE_IDS]
missile_vap_buf    -- contains the static vehicle appearance PDU information for
                    missiles.
missiles_initd
num_missiles_flying

```

2.1.1.3.1.31.1 network_missiles_init

This routine initializes the missile appearance packet structures to zero.

Internal Variables		
Variable	Type	Where Typedef Declared
i	int	Standard
vap	pointer to VehicleAppearanceVariant	p_sim.h

Table 2.1-33: network_missiles_init Information.

2.1.1.3.1.31.2 network_send_missile_appearance

This routine sends a missile's appearance packet. When a vehicle fires a missile, the network is sent separate appearance packets for the vehicle and the missile. The missile is treated as a separate vehicle by the network.

Parameters		
Parameter	Type	Where Typedef Declared
vehicle	unsigned short	Standard
chord_start	VECTOR	sim_types.h
chord_end	VECTOR	sim_types.h
orientation	T_MATRIX	sim_types.h
velocity	VECTOR	sim_types.h
guises	pointer to VehicleGuises	basic.h
Internal Variables		
Variable	Type	Where Typedef Declared
vap	register pointer to VehicleAppearanceVariant	p_sim.h
curr_miss	register pointer to MISSILE INFO	
x diff	float	Standard
y diff	float	Standard
z diff	float	Standard
Return Values		
Return Value	Type	Meaning
FALSE	int	either the missile appearance was not initialized, the missile's rotation vector is invalid, or it has an invalid missile vehicle id
TRUE	int	appearance packet was sent
Calls		
Function	Where Described	
vec_copy	Section 2.6.2.59.1	
d2f_mat_copy	Section 2.6.2.1.1	
net_current_time	See MCC CSCI Section 2.20.2.8.3	
network_get_net_handle	Section 2.1.1.3.2.12.1	
fmat_check	Section 2.6.2.11.1	
vec_dump	Section 2.6.2.60.1	
dump_core	Section 2.6.4.13.1	
network_fill_hdr_send_sim_pkt	Section 2.1.1.3.1.42.5	
PRO_SIM_APPEARANCE_SIZE	p_size.h	

Table 2.1-34: network_send_missile_appearance Information.

2.1.1.3.1.31.3 network_stop_missile_flyout

This routine is called to deactivate a missile when it is no longer flying out. The missile must have been initialized and flown out in order to be deactivated. The missile may have stopped flying out either because it impacted or because it flew out of the terrain database range. The routine checks to ensure that the missile has not already been deactivated, and then sends a Deactivate PDU over the network to inform the world that the missile is gone. Once the missile is deactivated, its appearance packet structure is reinitialized.

Parameters		
Parameter	Type	Where Typedef Declared
vehicle	unsigned short	Standard
Internal Variables		
Variable	Type	Where Typedef Declared
pdu	SimulationPDU	p_sim.h
pkt	register pointer to DeactivateRequestVariant	p_sim.h
i	int	Standard
curr_miss	pointer to MISSILE_INFO	
Calls		
Function	Where Described	
network_fill_hdr_send_sim_pkt	Section 2.1.1.3.1.42.5	
PRO_SIM_DEACTIVATE_REQ_SIZE	p_size.h	

Table 2.1-35: network_stop_missile_flyout Information.

2.1.1.3.1.31.4 network_send_missile_fire_pkt

This routine sends a Fire PDU describing the firing of a missile, in order for the world to make the appropriate sound and visual effects.

Parameters		
Parameter	Type	Where Typedef Declared
eventID	EventID	basic.h
ammo_type	int	Standard
detonator_type	ObjectType	p_sim.h
quantity	unsigned short	Standard
rate	unsigned short	Standard
targetType	unsigned char	Standard
targetID	pointer to VehicleID	basic.h
muzzle	WorldCoordinates	basic.h
velocity	VelocityVector	basic.h
tube	int	Standard
Internal Variables		
Variable	Type	Where Typedef Declared
pdu	SimulationPDU	p_sim.h
pkt	register pointer to FireVariant	p_sim.h
i	register int	Standard
curr_miss	register pointer to MISSILE_INFO	
vehicle	int	Standard
Return Values		
Return Value	Type	Meaning
vehicle	int	the missile vehicle ID
-1	int	no missile available
Calls		
Function	Where Described	
map_get_network_type_from_ammo_entry	Section 2.6.11.2.3	
vec_copy	Section 2.6.2.59.1	
fvec_copy	Section 2.6.2.26.1	
network_fill_hdr_send_sim_pkt	Section 2.1.1.3.1.42.5	
PRO_SIM_FIRE_SIZE	p_size.h	

Table 2.1-36: network_send_missile_fire_pkt Information.

2.1.1.3.1.32 net_xmit.c (./simnet/release/src/vehicle/libsrc/libSendNet/net_xmit.c)

Includes:

```
"send_loc.h"
"veh_app_loc.h"
"pro_sim.h"
"pro_size.h"
"pro_data.h"
"pro_mgmt.h"
"pro_timers.h"
"veh_type.h"
"libfail.h"
"libhull.h"
```

Declarations:

```
force_sending_veh_status    -- set equal to FALSE
```

Defines:

```
VEH_STATUS_MODE
EQUIP_STATUS_MODE
XMIT_OTHER
```

2.1.1.3.1.32.1 need_to_send_veh_status

This routine is called in order to send a Vehicle Status PDU, regardless of the timing. For example, "m1_ammoc.c" calls this routine when ammunition quantities change.

2.1.1.3.1.32.2 network_xmit

This routine sends out three types of packets at regular intervals specified by the protocol. The routine determines when to send out these appearance packets, equipment status packets and simulation status packets, and then sends them.

Internal Variables		
Variable	Type	Where Typedef Declared
time since last other	int	Standard
xmit_state	short	Standard
Calls		
Function	Where Described	
format vehicle appearance	Section 2.1.1.3.1.4.1	
network check veh appearance	Section 2.1.1.3.1.66.4	
send vehicle status	Section 2.1.1.3.1.71.1	
send equipment status		

Table 2.1-37: network_xmit Information.

2.1.1.3.1.32.3 network_xmit_idle

This routine determines when to transmit equipment status packets that must be sent when the simulation is in idle state, or is uninitialized. The packets are sent at a rate specified in "pro_timers.h".

Internal Variables		
Variable	Type	Where Typedef Declared
counter	int	Standard
Calls		
Function	Where Described	
send equipment status		

Table 2.1-38: network_xmit_idle Information.

2.1.1.3.1.33 non_impact.c

(./simnet/release/src/vehicle/libsrc/libSendNet/non_impact.c)

Includes:

```
"send_loc.h"
"pro_sim.h"
"pro_size.h"
"mun_type.h"
"sim_ammo.h"
"veh_app_loc.h"
"libmap.h"
"basic.h"
```

Declarations:

```
null_vehicleID
null_vehicle_coords
null_world_coords
```

2.1.1.3.1.33.1 network_send_non_impact

This routine sends an Impact PDU describing a non-impact.

Parameters		
Parameter	Type	Where Typedef Declared
eventID	EventID	basic.h
ammo_type	int	Standard
Internal Variables		
Variable	Type	Where Typedef Declared
pdu	SimulationPDU	p_sim.h
pkt	register pointer to ImpactVariant	p_sim.h
Calls		
Function	Where Described	
map_get_network_type_from_ammo_entry	Section 2.6.11.2.3	
vec_copy	Section 2.6.2.59.1	
ivec_copy	Section 2.6.2.26.1	
network_fill_hdr_send_sim_pkt	Section 2.1.1.3.1.42.5	
PRO_SIM_IMPACT_SIZE	p_size.h	

Table 2.1-39: network_send_non_impact Information.

2.1.1.3.1.34 nprintf.c

(./simnet/release/src/vehicle/libsrc/libSendNet/nprintf.c)

Includes:

"stdio.h"
 "varargs.h"
 "send_loc.h"
 "pro_assoc.h"
 "pro_mgmt.h"
 "pro_size.h"

2.1.1.3.1.34.1 nprintf

This utility routine creates a function similar to the standard C function "printf" which, instead of printing to the screen, sends an Error PDU out to the network under the Management Protocol. This routine uses the standard C VARARG capability for handling variable argument lists.

Parameters		
Parameter	Type	Where Typedef Declared
va_alist	va_dcl	Standard
Internal Variables		
Variable	Type	Where Typedef Declared
fmt	pointer to char	Standard
ap	va_list	Standard
buf[maxNetworkPDUSize / sizeof(long)]	long	Standard
pdu	pointer to ManagementPDU	p_mgmt.h
pkt	register pointer to ErrorReportVariant	p_mgmt.h
text	pointer to char	Standard
apdu	pointer to AssociationPDU	p_assoc.h
max_size	int	Standard
buf_size	int	Standard
malloc()	extern pointer to char	Standard
Calls		
Function	Where Described	
PRO ASSOC DATAGRAM HDR SIZE	p_size.h	
PRO MGMT ERROR REPORT SIZE	p_size.h	
format decoder	Section 2.6.4.15.3	
network fill_hdr_send_mgmt_pkt	Section 2.1.1.3.1.42.7	
PRO MGMT ERROR REPORT SIZE	p_size.h	

Table 2.1-40: nprintf Information.

2.1.1.3.1.35 position.c

(/simnet/release/src/vehicle/libsrc/libSendNet/position.c)

Code in this file is not compiled in the Version 6.6 release.

2.1.1.3.1.36 power_supply.c

(/simnet/release/src/vehicle/libsrc/libSendNet/power_supply.c)

Code in this file is not compiled in the Version 6.6 release.

2.1.1.3.1.37 reloadReq.c

(/simnet/release/src/vehicle/libsrc/libSendNet/reloadReq.c)

Code in this file is not compiled in the Version 6.6 release.

2.1.1.3.1.38 repaired.c

(/simnet/release/src/vehicle/libsrc/libSendNet/repaired.c)

Includes:

```
"send_loc.h"
"veh_app_loc.h"
"pro_assoc.h"
"pro_sim.h"
"pro_size.h"
```

2.1.1.3.1.38.1 send_repaired_pkt

This routine sends a transaction response to a Repair Request PDU.

Parameters		
Parameter	Type	Where Typedef Declared
supplierID	pointer to VehicleID	basic.h
result	RepairResult	p_sim.h
originator	pointer to SimulationAddress	address.h
tid	TransactionIdentifier	address.h
Internal Variables		
Variable	Type	Where Typedef Declared
pdu	SimulationPDU	p_sim.h
pkt	register pointer to RepairResponseVariant	p_sim.h
Calls		
Function	Where Described	
network fill hdr send sim rsp	Section 2.1.1.3.1.46.1	
PRO SIM REPAIR RESPONSE SIZE	p_size.h	

Table 2.1-41: send_repaired_pkt Information.

2.1.1.3.1.39 resupp_cancel.c

(./simnet/release/src/vehicle/libsrc/libSendNet/resupp_cancel.c)

This file is not used in the version 6.6 release.

2.1.1.3.1.40 resupp_offer.c

(./simnet/release/src/vehicle/libsrc/libSendNet/resupp_offer.c)

Includes:

"send_loc.h"
 "veh_app_loc.h"
 "pro_sim.h"
 "pro_size.h"
 "mun_type.h"

2.1.1.3.1.40.1 network_send_offer_packet

This routine sends a Resupply Offer PDU using the underlying datagram service.

Parameters		
Parameter	Type	Where Typedef Declared
receiverID	pointer to VehicleID	basic.h
num munitions	unsigned char	Standard
munitions	pointer to unsigned char	Standard
Internal Variables		
Variable	Type	Where Typedef Declared
pdu	SimulationPDU	p_sim.h
pkt	register pointer to ResupplyVariant	p_sim.h
i	register int	Standard
Calls		
Function	Where Described	
network_get_vehicle_type	Section 2.1.1.3.1.23.1	
network_get_simulator_type	Section 2.1.1.3.1.19.1	
network_fill_hdr_send_sim_pkt	Section 2.1.1.3.1.42.5	
PRO SIM RESUPPLY OFF SIZE	p_size.h	

Table 2.1-42: network_send_offer_packet Information.

2.1.1.3.1.41 resupp_recvd.c

(./simnet/release/src/vehicle/libsrc/libSendNet/resupp_recvd.c)

Includes:

"send_loc.h"
 "veh_app_loc.h"
 "mun_type.h"
 "pro_sim.h"
 "pro_size.h"

2.1.1.3.1.41.1 network_send_thank_you_packet

This routine sends a Resupply Received PDU using the underlying datagram service.

Parameters		
Parameter	Type	Where Typedef Declared
supplierID	pointer to VehicleID	basic.h
num munitions	unsigned char	Standard
munitions	register pointer to MunitionQuantity	basic.h
tid	TransactionIdentifier	address.h
Internal Variables		
Variable	Type	Where Typedef Declared
pdu	SimulationPDU	p_sim.h
pkt	register pointer to ResupplyVariant	p_sim.h
i	register int	Standard
Calls		
Function	Where Described	
network_get_vehicle_type	Section 2.1.1.3.1.23.1	
network_get_simulator_type	Section 2.1.1.3.1.19.1	
network_fill_hdr_send_sim_pkt	Section 2.1.1.3.1.42.5	
PRO SIM RESUPPLY RECEIVED SIZE	p_size.h	

Table 2.1-43: network_send_thank_you_packet Information.

2.1.1.3.1.42 send_dg_pkt.c

(./simnet/release/src/vehicle/libsrc/libSendNet/send_dg_pkt.c)

This file contains the routines that fill PDU headers for datagrams and that call the net interface to send them out.

Includes:

```
"stdio.h"
"pro_sim.h"
"pro_data.h"
"pro_mgmt.h"
"p_ivis.h"
"send_loc.h"
"veh_app_loc.h"
"net_stats.h"
```

2.1.1.3.1.42.1 fill_simHdr

This routine fills the header for Simulation PDUs.

Parameters		
Parameter	Type	Where Typedef Declared
pdu	pointer to SimulationPDU	p_sim.h
pduKind	SimulationPDUKind	p_sim.h

Table 2.1-44: fill_simHdr Information.

2.1.1.3.1.42.2 fill_mgmtHdr

This routine fills the header for Management PDUs.

Parameters		
Parameter	Type	Where Typedef Declared
pdu	pointer to ManagementPDU	p_mgmt.h
pduKind	ManagementPDUKind	p_mgmt.h

Table 2.1-45: fill_mgmtHdr Information.

2.1.1.3.1.42.3 fill_ivisHdr

Code in this routine is not used in the Version 6.6 release.

2.1.1.3.1.42.4 fill_dcHdr

This routine fills the header for Data Collection PDUs.

Parameters		
Parameter	Type	Where Typedef Declared
pdu	pointer to DataCollectionPDU	p_data.h
pduKind	DataCollectionPDUKind	p_data.h

Table 2.1-46: fill_dcHdr Information.

2.1.1.3.1.42.5 network_fill_hdr_send_sim_pkt

This routine sends out Simulation Protocol datagrams. It calls the routine **fill_simHdr()** to fill the packet header, and then calls the network interface to send out the packet.

Parameters		
Parameter	Type	Where Typedef Declared
pdu	pointer to SimulationPDU	p_sim.h
pduSize	int	Standard
pduKind	SimulationPDUKind	p_sim.h
Calls		
Function	Where Described	
fill_simHdr	Section 2.1.1.3.1.42.1	
AssocSendDatagram	See MCC CSCI Section 2.20.1.2.1	
network_get_net_handle	Section 2.1.1.3.2.12.1	
NOTE SENT		

Table 2.1-47: network_fill_hdr_send_sim_pkt Information.

2.1.1.3.1.42.6 network_fill_hdr_send_dc_pkt

This routine sends out Data Collection Protocol datagrams. It calls the routine **fill_dcHdr()** to fill the packet header, and then calls the network interface to send out the packet.

Parameters		
Parameter	Type	Where Typedef Declared
pdu	pointer to DataCollectionPDU	p_data.h
pduSize	int	Standard
pduKind	DataCollectionPDUKind	p_data.h
Calls		
Function	Where Described	
fill_dcHdr	Section 2.1.1.3.1.42.4	
AssocSendDatagram	See MCC CSCI Section 2.20.1.2.1	
network_get_net_handle	Section 2.1.1.3.2.12.1	
NOTE SENT		

Table 2.1-48: network_fill_hdr_send_dc_pkt Information.

2.1.1.3.1.42.7 network_fill_hdr_send_mgmt_pkt

This routine sends out Management Protocol datagrams. It calls the routine **fill_mgmtHdr()** to fill the packet header, and then calls the network interface to send out the packet.

Parameters		
Parameter	Type	Where Typedef Declared
pdu	pointer to ManagementPDU	p_data.h
pduSize	int	Standard
pduKind	ManagementPDUKind	p_mgmt.h
Calls		
Function	Where Described	
fill_mgmtHdr	Section 2.1.1.3.1.42.2	
AssocSendDatagram	See MCC CSCI Section 2.20.1.2.1	
network_get_net_handle	Section 2.1.1.3.2.12.1	
NOTE SENT		

Table 2.1-49: network_fill_hdr_send_mgmt_pkt Information.

2.1.1.3.1.42.8 network_fill_hdr_send_ivis_pkt

Code in this routine is not used in the Version 6.6 release.

2.1.1.3.1.43 send_loc.c

(./simnet/release/src/vehicle/libsrc/libSendNet/send_loc.c)

This file contains data definitions for local data structures.

Includes:

```
"sim_types.h"
"sim_dfns.h"
"send_loc.h"
"net_stats.h"
"net/network.h"
"pro_sim.h"
"veh_type.h"
```

Declarations:

```
netxmt_failed
packets_sent[maxProtocolFamily+1][maxPacketsPerFamily]
features          -- used to set special features such as tow down
veh_app_buf
init_buf
app_pkt           -- declared externally in "veh_app_loc.h"
init_pkt          -- declared externally in "veh_app_loc.h"
my_simulator_type -- describes the type of simulator (i.e., an M1 simulator or an
                  M2 simulator)
```

2.1.1.3.1.44 send_loc.h

(/simnet/release/src/vehicle/libsrc/libSendNet/send_loc.h)

This file contains local only #defines and declarations.

2.1.1.3.1.45 send_pt_pkt.c

(/simnet/release/src/vehicle/libsrc/libSendNet/send_pt_pkt.c)

Includes:

```
"stdio.h"
"basic.h"
"pro_assoc.h"
"p2p.h"
"send_loc.h"
```

2.1.1.3.1.45.1 send_pt_packet

This routine sends a point to point packet.

Parameters		
Parameter	Type	Where Typedef Declared
buf	pointer to char	Standard
pduSize	int	Standard
exerciseID	ExerciseID	basic.h
protocol	AssociationUserProtocol	p_assoc.h
addr	pointer to SimulationAddress	address.h
Return Values		
Return Value	Type	Meaning
FALSE	int	either not using ethernet or transmission failed
TRUE	int	procedure was successful
Calls		
Function	Where Described	
PointToPointSendPDU	Section 2.1.1.2.2.2.1	
network get net handle	Section 2.1.1.3.2.12.1	

Table 2.1-50: send_pt_packet Information.

2.1.1.3.1.46 send_rsp.c

(./simnet/release/src/vehicle/libsrc/libSendNet/send_rsp.c)

This file contains routines for sending responses to transaction requests.

Includes:

```
"stdio.h"
"sim_dfns.h"
"pro_assoc.h"
"pro_sim.h"
"pro_data.h"
"pro_num.h"
"net_stats.h"
```

2.1.1.3.1.46.1 network_fill_hdr_send_sim_rsp

This routine fills the PDU headers for Simulation Protocol transaction responses, and calls the network interfaces to send them out.

Parameters		
Parameter	Type	Where Typedef Declared
pdu	register pointer to SimulationPDU	p_sim.h
size	int	Standard
kind	SimulationPDUKind	p_sim.h
originator	pointer to SimulationAddress	address.h
tid	TransactionIdentifier	address.h
cache	int	Standard
Calls		
Function	Where Described	
fill simHdr	Section 2.1.1.3.1.42.1	
AssocSendResponse	See MCC CSCI Section 2.20.1.4.2	
network_get_net_handle	Section 2.1.1.3.2.12.1	
NOTE SENT		

Table 2.1-51: network_fill_hdr_send_sim_rsp Information.

2.1.1.3.1.46.2 network_fill_hdr_send_dc_rsp

This routine fills the PDU headers for Data Collection Protocol transaction responses, and calls the network interfaces to send them out.

Parameters		
Parameter	Type	Where Typedef Declared
pdu	register pointer to DataCollectionPDU	p_data.h
size	int	Standard
kind	DataCollectionPDUKind	p_data.h
originator	pointer to SimulationAddress	address.h
cache	int	Standard
Calls		
Function	Where Described	
fill_dcHdr	Section 2.1.1.3.1.42.4	
AssocSendResponse	See MCC CSCI Section 2.20.1.4.2	
network_get_net_handle	Section 2.1.1.3.2.12.1	
NOTE SENT		

Table 2.1-52: network_fill_hdr_send_dc_rsp Information.

2.1.1.3.1.47 send_trans.c

(./simnet/release/src/vehicle/libsrc/libSendNet/send_trans.c)

This file contains routines for sending transactions.

Includes:

```
"stdio.h"
"sim_dfns.h"
"pro_assoc.h"
"pro_sim.h"
"pro_data.h"
"pro_num.h"
```

2.1.1.3.1.47.1 network_fill_hdr_send_sim_trans

This routine fills the PDU headers for Simulation Protocol transactions, and calls the network interface to send them out.

Parameters		
Parameter	Type	Where Typedef Declared
pdu	register pointer to SimulationPDU	p_sim.h
size	int	Standard
kind	SimulationPDUKind	p_sim.h
respondent	pointer to SimulationAddress	address.h
(callback)()	pointer to int	Standard
cparam	pointer to long int	Standard
(timeout)()	pointer to int	Standard
tparam	pointer to long int	Standard
Return Values		
Return Value	Type	Meaning
FALSE	int	the transaction failed
TRUE	int	the transaction was successful
Calls		
Function	Where Described	
fill_simHdr	Section 2.1.1.3.1.42.1	
AssocSendTransact	See MCC CSCI Section 2.20.1.4.1	
network_get_net handle	Section 2.1.1.3.2.12.1	

Table 2.1-53: network_fill_hdr_send_sim_trans Information.

2.1.1.3.1.47.2 network_fill_hdr_send_dc_trans

This routine fills the PDU headers for Data Collection Protocol transactions, and calls the network interface to send them out.

Parameters		
Parameter	Type	Where Typedef Declared
pdu	register pointer to DataCollectionPDU	p_data.h
size	int	Standard
kind	DataCollectionPDUKind	p_data.h
respondent	pointer to SimulationAddress	address.h
(callback)()	pointer to int	Standard
cparam	pointer to long int	Standard
(timeout)()	pointer to int	Standard
tparam	pointer to long int	Standard
Return Values		
Return Value	Type	Meaning
FALSE	int	the transaction failed
TRUE	int	the transaction was successful
Calls		
Function	Where Described	
fill dcHdr	Section 2.1.1.3.1.42.4	
AssocSendTransact	See MCC CSCI Section 2.20.1.4.1	
network get net handle	Section 2.1.1.3.2.12.1	

Table 2.1-54: network_fill_hdr_send_dc_trans Information.

2.1.1.3.1.48 service_req.c

(/simnet/release/src/vehicle/libsrc/libSendNet/service_req.c)

Includes:

"send_loc.h"
 "veh_app_loc.h"
 "pro_sim.h"
 "mun_type.h"

2.1.1.3.1.48.1 network_send_feed_me_packet

This routine sends a Service Request PDU using the underlying datagram service.

Parameters		
Parameter	Type	Where Typedef Declared
supplierID	pointer to VehicleID	basic.h
num_munitions	unsigned char	Standard
munitions	register pointer to MunitionQuantity	basic.h
Internal Variables		
Variable	Type	Where Typedef Declared
pdu	SimulationPDU	p_sim.h
pkt	register pointer to ResupplyVariant	p_sim.h
i	register int	Standard
Calls		
Function	Where Described	
network get vehicle type	Section 2.1.1.3.1.23.1	
network get simulator type	Section 2.1.1.3.1.19.1	
network fill_hdr_send_sim_pkt	Section 2.1.1.3.1.42.5	
PRO SIM SERVICE REQ SIZE	p_size.h	

Table 2.1-55: network_send_feed_me_packet Information.

2.1.1.3.1.49 set_ex_id.c

(./simnet/release/src/vehicle/libsrc/libSendNet/set_ex_id.c)

Includes:

"send_loc.h"

"veh_app_loc.h"

2.1.1.3.1.49.1 network_set_exercise_id

This routine changes the exercise ID stored in the Vehicle Appearance PDU to *new_id*.

Parameters		
Parameter	Type	Where Typedef Declared
new_id	ExerciseID	basic.h

Table 2.1-56: network_set_exercise_id Information.

2.1.1.3.1.50 set_force.c

(./simnet/release/src/vehicle/libsrc/libSendNet/set_force.c)

Includes:

"send_loc.h"

"veh_app_loc.h"

2.1.1.3.1.50.1 network_set_force

This routine changes the force ID stored in the Vehicle Appearance PDU to *force*.

Parameters		
Parameter	Type	Where Typedef Declared
force	ForceID	basic.h

Table 2.1-57: network_set_force Information.

2.1.1.3.1.51 set_guises.c

(./simnet/release/src/vehicle/libsrc/libSendNet/set_guises.c)

Includes:

"pro_sim.n"

"send_loc.h"

"veh_app_loc.h"

2.1.1.3.1.51.1 network_set_vehicle_guises

This routine changes the vehicle's disguises stored in the Vehicle Appearance PDU to *guises*.

Parameters		
Parameter	Type	Where Typedef Declared
guises	pointer to VehicleGuises	basic.h

Table 2.1-58: network_set_vehicle_guises Information.

2.1.1.3.1.52 set_sim_type.c

(./simnet/release/src/vehicle/libsrc/libSendNet/set_sim_type.c)

Includes:

"send_loc.h"

"veh_app_loc.h"

2.1.1.3.1.52.1 network_set_simulator_type

This routine changes the simulator to the type passed in *type*.

Parameters		
Parameter	Type	Where Typedef Declared
<i>type</i>	SimulatorType	basic.h

Table 2.1-59: network_set_simulator_type Information.

2.1.1.3.1.53 set_veh_app.c

(./simnet/release/src/vehicle/libsrc/libSendNet/set_veh_app.c)

Includes:

"pro_sim.h"

"send_loc.h"

"veh_app_loc.h"

2.1.1.3.1.53.1 network_set_vehicle_appearance

This routine changes the vehicle's appearance stored in the Vehicle Appearance PDU to *new_appearance*.

Parameters		
Parameter	Type	Where Typedef Declared
<i>new_appearance</i>	unsigned long	Standard

Table 2.1-60: network_set_vehicle_appearance Information.

2.1.1.3.1.54 set_veh_clas.c

(./simnet/release/src/vehicle/libsrc/libSendNet/set_veh_clas.c)

Includes:

```
"stdio.h"
"send_loc.h"
"veh_app_loc.h"
"mass_stdh.h"
"dgi_stdg.h"
"sim_cig_if.h"
"libmain.h"
```

2.1.1.3.1.54.1 network_set_vehicle_class

This routine changes the vehicle's class stored in the Vehicle Appearance PDU to *new_class*.

Parameters		
Parameter	Type	Where Typedef Declared
new_class	unsigned char	Standard

Table 2.1-61: network_set_vehicle_class Information.

2.1.1.3.1.55 set_veh_id.c

(./simnet/release/src/vehicle/libsrc/libSendNet/set_veh_id.c)

Includes:

```
"pro_sim.h"
"send_loc.h"
"veh_app_loc.h"
```

2.1.1.3.1.55.1 network_set_vehicle_id

This routine sets the vehicle's ID number stored in the Vehicle Appearance PDU to *new_id*.

Parameters		
Parameter	Type	Where Typedef Declared
new_id	pointer to VehicleID	basic.h
Called By		
Function	Where Described	
network_init	Section 2.1.1.3.2.12.3	

Table 2.1-62: network_set_vehicle_id information.

2.1.1.3.1.56 set_xmt_fail.c

(./simnet/release/src/vehicle/libsrc/libSendNet/set_xmt_fail.c)

Includes:

"send_loc.h"

2.1.1.3.1.56.1 set_xmt_failed

This routine sets the *netxmt_failed* flag to either TRUE or FALSE. It can be used during debugging to check whether network transmissions are working.

Parameters		
Parameter	Type	Where Typedef Declared
state	BOOLEAN	sim_types.h

Table 2.1-63: set_xmt_failed Information.

2.1.1.3.1.57 shell_fire.c

(./simnet/release/src/vehicle/libsrc/libSendNet/shell_fire.c)

Includes:

"send_loc.h"
 "veh_app_loc.h"
 "mun_type.h"
 "pro_sim.h"
 "pro_size.h"
 "sim_ammo.h"
 "libmap.h"

Declarations:

null_vehicleID

2.1.1.3.1.57.1 network_send_shell_fire_pkt

This routine sends a Fire PDU describing the firing of a shell, in order for the world to make the appropriate sound and visual effects.

Parameters		
Parameter	Type	Where Typedef Declared
eventID	EventID	basic.h
ammo_type	int	Standard
detonator_type	ObjectType	p_sim.h
quantity	unsigned short	Standard
rate	unsigned short	Standard
targetType	unsigned short	Standard
targetID	pointer to VehicleID	basic.h
muzzle	WorldCoordinates	basic.h
velocity	VelocityVector	basic.h
range	float	Standard
slew_rate	float	Standard
ammoSelected	ObjectType	p_sim.h
Internal Variables		
Variable	Type	Where Typedef Declared
pdu	SimulationPDU	p_sim.h
pkt	register pointer to FireVariant	p_sim.h
Calls		
Function	Where Described	
map_get_network_type_from_ammo_entry	Section 2.6.11.2.3	
vec copy	Section 2.6.2.59.1	
fvec copy	Section 2.6.2.26.1	
network_fill_hdr_send_sim_pkt	Section 2.1.1.3.1.42.5	
PRO_SIM_FIRE_SIZE	p_size.h	

Table 2.1-64: network_send_shell_fire_pkt Information.

2.1.1.3.1.58 show_effect.c

(./simnet/release/src/vehicle/libsrc/libSendNet/show_effect.c)

Code in this file is not compiled in the Version 6.6 release.

2.1.1.3.1.59 sim_status.c

(./simnet/release/src/vehicle/libsrc/libSendNet/sim_status.c)

Includes:

```
"sys/type.h"
"sys/timeb.h"
"send_loc.h"
"veh_app_loc.h"
"pro_data.h"
"pro_assoc.h"
"pro_size.h"
"pro_num.h"
"net_stats.h"
```

2.1.1.3.1.59.1 send_simulation_status_pkt

This routine sends a Simulation Status PDU using the datagram service.

Parameters		
Parameter	Type	Where Typedef Declared
exercise	ExerciseID	basic.h
Internal Variables		
Variable	Type	Where Typedef Declared
pdu	DataCollectionPDU	p_data.h
pkt	register pointer to SimulationStatusVariant	p_data.h
current time	struct timeb	Standard
elapsed time	long	Standard
Calls		
Function	Where Described	
network_get_simulator_type	Section 2.1.1.3.1.19.1	
timers_elapsed_milliseconds	Section 2.6.3.10.1	
AssocSendDatagram	See MCC CSCI Section 2.20.1.2.1	
network_get_net_handle	Section 2.1.1.3.2.12.1	
PRO_DATA_SIMULATION_STATUS_SIZE	p_data.h	
NOTE SENT		

Table 2.1-65: send_simulation_status_pkt Information.

2.1.1.3.1.59.2 send_simulation_status_trans

This routine sends a Simulation Status PDU using the transaction service.

Parameters		
Parameter	Type	Where Typedef Declared
originator	pointer to SimulationAddress	address.h
tid	TransactionIdentifier	address.h
exercise	ExerciseID	basic.h
Internal Variables		
Variable	Type	Where Typedef Declared
pdu	DataCollectionPDU	p_data.h
pkt	register pointer to SimulationStatusVariant	p_data.h
elapsed time	long	Standard
current time	struct timeb	Standard
Calls		
Function	Where Described	
timers elapsed milliseconds	Section 2.6.3.10.1	
AssocSendTransact	See MCC CSCI Section 2.20.1.4.1	
network_get_net_handle	Section 2.1.1.3.2.12.1	
PRO DATA SIMULATION STATUS SIZE	p_size.h	
NOTE SENT		

Table 2.1-66: send_simulation_status_trans Information.

2.1.1.3.1.60 spec_appear.c

(./simnet/release/src/vehicle/libsrc/libSendNet/spec_appear.c)

The routine in this file is stubbed in anticipation of a later release.

2.1.1.3.1.61 spec_status.c

(./simnet/release/src/vehicle/libsrc/libSendNet/spec_status.c)

The routine in this file is stubbed in anticipation of a later release.

2.1.1.3.1.62 stat_change.c

(./simnet/release/src/vehicle/libsrc/libSendNet/stat_change.c)

Includes:

"send_loc.h"
 "veh_app_loc.h"
 "pro_data.h"
 "pro_size.h"

2.1.1.3.1.62.1 network_send_status_change

This routine sends a Status Change PDU to notify the world that a change in vehicle status has occurred. The routine assumes that all bits are set for VehicleSubsystemStatus in a global place in memory. The change in vehicle status may be due to the vehicle being destroyed or recreated, or to vehicle subsystems being damaged or repaired.

Parameters		
Parameter	Type	Where Typedef Declared
effect	StatusChangeEffect	p_data.h
cause	int	Standard
agentID	pointer to VehicleID	basic.h
eventID	EventID	basic.h
subsystem	pointer to VehicleSubsystems	status.h
Internal Variables		
Variable	Type	Where Typedef Declared
pdu	DataCollectionPDU	p_data.h
pkt	register pointer to StatusChangeVariant	p_data.h
Calls		
Function	Where Described	
network_fill_hdr_send_dc_pkt	Section 2.1.1.3.1.42.6	
PRO_DATA_STATUS_CHANGE_SIZE	p_size.h	

Table 2.1-67: network_send_status_change Information.

2.1.1.3.1.63 stat_rsp.c

(./simnet/release/src/vehicle/libsrc/libSendNet/stat_rsp.c)

Includes:

"stdio.h"
 "sim_macros.h"
 "send_loc.h"
 "veh_app_loc.h"
 "pro_data.h"
 "pro_num.h"
 "pro_assoc.h"
 "pro_size.h"
 "assoc.h"
 "libnetwork.h"
 "net_stats.h"

Declarations:

vehIrrelevant
 send_status_response_trans()
 send_status_in_f__ing_multicast_group_zero()

2.1.1.3.1.63.1 network_respond_to_query_trans

This routine sends a transaction response to a status query.

Parameters		
Parameter	Type	Where Typedef Declared
pkt	pointer to StatusQueryVariant	p_data.h
exercise	ExerciseID	basic.h
originator	pointer to SimulationAddress	address.h
tid	TransactionIdentifier	address.h
Internal Variables		
Variable	Type	Where Typedef Declared
ignore	int	Standard
pdu	DataCollectionPDU	p_data.h
rsp	pointer to StatusResponseVariant	p_data.h
Calls		
Function	Where Described	
can ignore	Section 2.1.1.3.1.63.3	
send vehicle status trans	Section 2.1.1.3.1.71.3	
send exercise status trans	Section 2.1.1.3.1.14.2	
send simulation status trans	Section 2.1.1.3.1.59.2	
send status response trans	Section 2.1.1.3.1.63.7	

Table 2.1-68: network_respond_to_query_trans Information.

2.1.1.3.1.63.2 network_respond_to_query_pkt

This routine sends a datagram response to a status query.

Parameters		
Parameter	Type	Where Typedef Declared
pkt	pointer to StatusQueryVariant	p_data.h
exercise	ExerciseID	basic.h
Calls		
Function	Where Described	
can ignore	Section 2.1.1.3.1.63.3	
send_vehicle_status_in_f__ing_ multicast_group_zero	Section 2.1.1.3.1.71.2	
send_vehicle_status	Section 2.1.1.3.1.71.1	
send_exercise_status_pkt	Section 2.1.1.3.1.14.1	
send_simulation_status_pkt	Section 2.1.1.3.1.59.1	

Table 2.1-69: network_respond_to_query_pkt Information.

2.1.1.3.1.63.3 can_ignore

The status query unitRelation field allows the query to select respondents based on the organizational units they simulate. Each receiver of the query must determine whether it is necessary to respond. This routine determines whether the invoking process is within the unit relationship specified by the status query packet.

Parameters		
Parameter	Type	Where Typedef Declared
pkt	pointer to StatusQueryVariant	p_data.h
exercise	ExerciseID	basic.h
Internal Variables		
Variable	Type	Where Typedef Declared
ourID	register pointer to VehicleID	basic.h
Return Values		
Return Value	Type	Meaning
0	int	must respond to the status query
-1	int	procedure has failed
1	int	can ignore the status query
Calls		
Function	Where Described	
network_get_exercise_id	Section 2.1.1.3.1.16.1	
network_get_vehicle_id	Section 2.1.1.3.1.22.1	
VEHICLE_IDS_EQUAL	sim_macros.h	
network_get_simulator_type	Section 2.1.1.3.1.19.1	
network_get_vehicle_force	Section 2.1.1.3.1.17.1	
network_get_vehicle_unit	Section 2.1.1.3.1.20.1	
same_unit	Section 2.1.1.3.1.63.4	
included_unit	Section 2.1.1.3.1.63.5	
including_unit	Section 2.1.1.3.1.63.6	

Table 2.1-70: can_ignore Information.

2.1.1.3.1.63.4 same_unit

This routine determines if the invoking process is in the same organizational unit as *other*.

Parameters		
Parameter	Type	Where Typedef Declared
<i>other</i>	pointer to <i>OrganizationalUnit</i>	basic.h
<i>me</i>	pointer to <i>OrganizationalUnit</i>	basic.h
Internal Variables		
Variable	Type	Where Typedef Declared
<i>op</i>	register pointer to <i>UnitIdentifier</i>	basic.h
<i>mp</i>	register pointer to <i>UnitIdentifier</i>	basic.h
<i>i</i>	register int	Standard
Return Values		
Return Value	Type	Meaning
FALSE	int	not in the same organizational unit as <i>other</i>
TRUE	int	in the same organziational unit as <i>other</i>

Table 2.1-71: same_unit Information.

2.1.1.3.1.63.5 included_unit

This routine determines whether the organizational unit of the invoking process is included by the organizational unit of *other*.

Parameters		
Parameter	Type	Where Typedef Declared
other	pointer to OrganizationalUnit	basic.h
me	pointer to OrganizationalUnit	basic.h
Internal Variables		
Variable	Type	Where Typedef Declared
op	register pointer to UnitIdentifier	basic.h
mp	register pointer to UnitIdentifier	basic.h
i	register int	Standard
last_valid_value	register int	Standard
Return Values		
Return Value	Type	Meaning
FALSE	int	we are not included by the organizational unit of <i>other</i>
TRUE	int	we are included by the organizational unit of <i>other</i>

Table 2.1-72: included_unit Information.

2.1.1.3.1.63.6 including_unit

This routine determines whether the organizational unit of *other* is included by the organizational unit of the invoking process.

Parameters		
Parameter	Type	Where Typedef Declared
other	pointer to OrganizationalUnit	basic.h
me	pointer to OrganizationalUnit	basic.h
Internal Variables		
Variable	Type	Where Typedef Declared
op	register pointer to UnitIdentifier	basic.h
mp	register pointer to UnitIdentifier	basic.h
i	register int	Standard
last valid value	register int	Standard
Return Values		
Return Value	Type	Meaning
FALSE	int	includes the organizational unit of <i>other</i> .
TRUE	int	does not include the organizational unit of <i>other</i> .

Table 2.1-73: including_unit Information.

2.1.1.3.1.63.7 send_status_response_trans

This routine sends a Status Response PDU as a transaction response to a Status Query PDU when the conditions specified in the Status Query PDU are not met.

Parameters		
Parameter	Type	Where Typedef Declared
result	register pointer to StatusResponseVariant	p_data.h
originator	pointer to SimulationAddress	address.h
tid	TransactionIdentifier	address.h
exercise	ExerciseID	basic.h
Internal Variables		
Variable	Type	Where Typedef Declared
pdu	DataCollectionPDU	p_data.h
r	register pointer to StatusResponseVariant	p_data.h
Calls		
Function	Where Described	
AssocSendResponse	See MCC CSCI Section 2.20.1.4.2	
network_get_net_handle	Section 2.1.1.3.2.12.1	
PRO DATA STATUS RESPONSE SIZE	p_size.h	
NOTE SENT		
Called By		
Function	Where Described	
network respond to query trans	Section 2.1.1.3.1.63.1	

Table 2.1-74: send_status_response_trans Information.

2.1.1.3.1.64 targetDiseng.c

(./simnet/release/src/vehicle/libsrc/libSendNet/targetDiseng.c)

Code in this file is not compiled in the Version 6.6 release.

2.1.1.3.1.65 target_engag.c

(./simnet7/release/src/vehicle/libsrc/libSendNet/target_engag.c)

Code in this file is not compiled in the Version 6.6 release.

2.1.1.3.1.66 thresh.c

(./simnet/release/src/vehicle/libsrc/libSendNet/thresh.c)

This file contains the routines which decide when to send out vehicle appearance packets. The vehicle specific code may substitute for the routines in this file. If substitutions are made, no references may be made to "thresh.c" routines in order to prevent duplicate reference problems with the linker.

Includes:

"send_loc.h"
"pro_sim.h"
"pro_size.h"
"veh_type.h"
"net/network.h"
"libapp.h"
"libkin.h"
"libhull.h"
"libmatrix.h"
"net_stats.h"

Declarations:

prev_update
thresholds
v_pkt_verbose
network_suppress_app

2.1.1.3.1.66.1 v_pkt_verbose_mode

This routine is called to force the simulation to send out a vehicle appearance packet every tick without checking the RVA algorithms or calling `network_check_veh_app`. The routine sets the variable `v_pkt_verbose` equal to TRUE.

2.1.1.3.1.66.2 network_stop_sending_app

This routine sets the variable `network_suppress_app` equal to TRUE.

2.1.1.3.1.66.3 network_restart_sending_app

This routine is used for debugging purposes. It sets the variable `network_suppress_app` equal to FALSE, preventing appearance packets from being sent on the network.

2.1.1.3.1.66.4 network_check_veh_appearance

This routine is called from `net_xmit()`. The routine decides whether to send out a vehicle appearance packet, and sends the packet, if necessary.

Parameters		
Parameter	Type	Where Typedef Declared
pdu	pointer to SimulationPDU	p_sim.h
Internal Variables		
Variable	Type	Where Typedef Declared
pkt	pointer to VehicleAppearanceVariant	p_sim.h
Calls		
Function	Where Described	
network_fill_hdr_send_sim_pkt	Section 2.1.1.3.1.42.5	
PRO_SIM_APPEARANCE_SIZE	p_size.h	
AppearanceDiscrepancyExceedsThresholds	Section 2.5.16.3.2	
timers_get_current_tick	Section 2.6.3.1.1	

Table 2.1-75: network_check_veh_appearance Information.

2.1.1.3.1.66.5 network_init_thresholds

This routine is called to initialize the thresholds used to determine when to send out Vehicle Appearance packets. Note that these are only initial values; they will be overwritten as soon as the first VAP is generated.

Parameters		
Parameter	Type	Where Typedef Declared
thresh_file	pointer to char	Standard
Return Values		
Return Value	Type	Meaning
ReadDiscrepancyThresholds (thresh_file, &thresholds)	int	thresholds
Calls		
Function	Where Described	
network_get_vehicle_type	Section 2.1.1.3.1.23.1	
fmat_ident_init	Section 2.6.2.13.1	

Table 2.1-76: network_init_thresholds Information.

2.1.1.3.1.67 tow_status.c

(./simnet/release/src/vehicle/libsrc/libSendNet/tow_status.c)

Includes:

"send_loc.h"
"veh_appear.h"

2.1.1.3.1.67.1 network_tow_launcher_up

This routine is used by the M2 to indicate that the TOW launcher is up.

2.1.1.3.1.67.2 network_tow_launcher_down

This routine is used by the M2 to indicate that the TOW launcher is down.

2.1.1.3.1.68 use_activ.c

(./simnet/release/src/vehicle/libsrc/libSendNet/use_activ.c)

Includes:

"ctype.h"
"send_loc.h"
"veh_app_loc.h"
"pro_sim.h"
"pro_assoc.h"
"libhull.h"
"libkin.h"
"libturret.h"
"librepair.h"
"libmain.h"
"libcig.h"

Defines: DB_FILENAME_SIZE

2.1.1.3.1.68.1 format_db_filename

This routine formats the database filename for activation.

Parameters		
Parameter	Type	Where Typedef Declared
name	pointer to char	Standard
database	pointer to char	Standard
version	int	Standard
Internal Variables		
Variable	Type	Where Typedef Declared
string[DB_FILENAME_SIZE]	char	Standard
s	pointer to char	Standard

Table 2.1-77: format_db_filename Information.

2.1.1.3.1.68.2 network_use_activation

This routine is called when `process_an_activate()`, located in `libRcvNet`, gets an activate packet. First, the activate packet is saved for future use. Then, information from the activate packet is moved into the vehicle appearance packet, `app_pkt`. Note that the vehicle specific code processes activation packets using the routine `network_process_activation_parameters()`. Note that `ammo_init()` and `fuel_init()` are called after the BCOPY of `init_pkt`, but before calling `network_process_activation_parameters()`. This allows vehicles with multiple configurations to know what they are simulating before determining their maximum fuel and ammunition capabilities.

Parameters		
Parameter	Type	Where Typedef Declared
pkt	pointer to ActivateRequestVariant	p_sim.h
originator	pointer to SimulationAddress	address.h
tid	TransactionIdentifier	address.h
Internal Variables		
Variable	Type	Where Typedef Declared
database[23]	char	Standard
yaw	REAL	sim_types.h
vap	pointer to VehicleAppearanceVariant	p_sim.h
Calls		
Function	Where Described	
ammo_init	Section 2.2.5.1.1	
fuel_init	Section 2.3.5.2.2	
format_db_filename	Section 2.1.1.63.1.68.1	
cig_use_database_named		
network_process_activation_parameters		
hull_uninit	Section 2.5.9.1.2	
repair_uninit	Section 2.5.4.19.2	
kinematics_pos_init	Section 2.5.8.13.1	
turret_pos_init	Section 2.5.5.2.2	
send_activate_response	Section 2.1.1.3.1.1.1	
activate_simulation	Section 2.5.1.1.3	

Table 2.1-78: network_use_activation Information.

2.1.1.3.1.69 veh_app_loc.h

(./simnet/release/src/vehicle/libsrc/libSendNet/veh_app_loc.h)

This file contains local declarations used in libSendNet.

2.1.1.3.1.70 veh_impact.c

(./simnet/release/src/vehicle/libsrc/libSendNet/veh_impact.c)

Includes:

"stdio.h"
"sim_types.h"
"sim_dfns.h"
"sim_macros.h"
"send_loc.h"
"pro_sim.h"
"pro_size.h"
"mun_type.h"
"veh_app_loc.h"
"sim_ammo.h"
"mass_stdh.h"
"dgi_stdg.h"
"sim_cig_if.h"
"librva.h"
"libmap.h"
"basic.h"

Defines: CIG_SAYS_TURRET

Declarations:

null_vehicleID
null_world_coordinates

2.1.1.3.1.70.1 network_send_vehicle_impact

This routine sends an Impact PDU transaction for a vehicle impact.

Parameters		
Parameter	Type	Where Typedef Declared
eventID	EventID	basic.h
ammo_type	int	Standard
detonator_type	ObjectType	p_sim.h
quantity	unsigned int	Standard
rate	unsigned int	Standard
vehicle_struck	pointer to VehicleID	basic.h
object tag	int	Standard
chord_start	pointer to VehicleCoordinates	basic.h
chord_end	pointer to VehicleCoordinates	basic.h
location	WorldCoordinates	basic.h
Internal Variables		
Variable	Type	Where Typedef Declared
pdu	SimulationPDU	p_sim.h
pkt	register pointer to ImpactVariant	p_sim.h
trajectory	VehicleCoordinates	basic.h
Calls		
Function	Where Described	
map_get_network_type_from_ammo_entry	Section 2.6.11.2.3	
vec copy	Section 2.6.2.59.1	
fvec copy	Section 2.6.2.26.1	
fvec sub	Section 2.6.2.29.1	
network_fill_hdr_send_sim_trans	Section 2.1.1.3.1.47.1	
PRO SIM IMPACT	p_size.h	

Table 2.1-79: network_send_vehicle_impact Information.

2.1.1.3.1.71 veh_status.c

(./simnet/release/src/vehicle/libsrc/libSendNet/veh_status.c)

Includes:

"send_loc.h"
 "veh_app_loc.h"
 "pro_assoc.h"
 "pro_data.h"
 "pro_size.h"
 "net_stats.h"

Declarations:

build_vehicle_status()

2.1.1.3.1.71.1 send_vehicle_status

This routine is called by **net_xmit()** or **process_query_pkt()** to send a Vehicle Status PDU using the datagram service. It calls **build_vehicle_status()** to fill in the packet.

Internal Variables		
Variable	Type	Where Typedef Declared
pdu	DataCollectionPDU	p_data.h
pkt	pointer to VehicleStatusVariant	p_data.h
Calls		
Function	Where Described	
build_vehicle_status	Section 2.1.1.3.1.71.4	
network_fill_hdr_send_dc_pkt	Section 2.1.1.3.1.42.6	
PRO_DATA_VEHICLE_STATUS_SIZE	p_size.h	

Table 2.1- 80: send_vehicle_status Information.

2.1.1.3.1.71.2 send_vehicle_status_in_f__ing_multicast_group_zero

This routine sends a Vehicle Status PDU transaction in multicast group zero.

Internal Variables		
Variable	Type	Where Typedef Declared
pdu	DataCollectionPDU	p_data.h
pkt	pointer to VehicleStatusVariant	p_data.h
Calls		
Function	Where Described	
build_vehicle_status	Section 2.1.1.3.1.71.4	
fill_dcHdr	Section 2.1.1.3.1.42.6	
AssocSendDatagram	See MCC CSCI Section 2.20.4.2.1	
network_get_net_handle	Section 2.1.1.3.2.12.1	
PRO DATA VEHICLE STATUS SIZE	p_size.h	
NOTE SENT		

Table 2.1-81: send_vehicle_status_in_f__ing_multicast_group_zero Information.

2.1.1.3.1.71.3 send_vehicle_status_trans

This routine sends a Vehicle Status PDU as a response to a Status Query transaction.

Parameters		
Parameter	Type	Where Typedef Declared
originator	pointer to SimulationAddress	address.h
tid	TransactionIdentifier	address.h
exercise	ExerciseID	basic.h
Internal Variables		
Variable	Type	Where Typedef Declared
pdu	DataCollectionPDU	p_data.h
pkt	pointer to VehicleStatusVariant	p_data.h
Calls		
Function	Where Described	
build_vehicle_status	Section 2.1.1.3.1.71.4	
fill_dcHdr	Section 2.1.1.3.1.42.4	
AssocSendResponse	See MCC CSCI Section 2.20.1.4.2	
network_get_net_handle	Section 2.1.1.3.2.12.1	
PRO DATA VEHICLE STATUS SIZE	p_size.h	
NOTE SENT		

Table 2.1-82: send_vehicle_status_trans Information.

2.1.1.3.1.71.4 build_vehicle_status

This routine fills in the fields of a Vehicle Status PDU. This routine fills in the vehicle independent fields and calls `fill_vehicle_spec_status()` to fill in the vehicle specific fields.

Parameters		
Parameter	Type	Where Typedef Declared
pkt	register pointer to VehicleStatusVariant	p_data.h
Calls		
Function	Where Described	
vehicle_get_elapsed km	Section 2.3.2.3.4.4	
fill_vehicle_spec_status	Section 2.2.7.1 Section 2.3.7.1 Section 2.4.6.1	

Table 2.1-83: build_vehicle_status Information.

2.1.1.3.2 libRcvNet

(./simnet/release/src/vehicle/libsrc/libRcvNet [libRcvNet])

Routines in LibRcvNet receive and process various types of SIMNET data packets.

2.1.1.3.2.1 activate.c

(./simnet/release/src/vehicle/libsrc/libRcvNet/activate.c)

Includes:

```
"rcv_loc.h"
"net_stats.h"
"pro_sim.h"
"pro_num.h"
"pro_size.h"
"pro_assoc.h"
"address.h"
"net/network.h"
"net/nettab.h"
"assoc.h"
```

2.1.1.3.2.1.1 process_activate_request

This routine is called to process an activate request PDU. The Activate PDU may be received either from the network or from the simulation console (if the -p option was specified). The routine calls `network_use_activation()`, located in libSendNet, to save the different parts of the PDU and to start the simulation.

Parameters		
Parameter	Type	Where Typedef Declared
p	pointer to ActivateRequestVariant	p_sim.h
originator	pointer to SimulationAddress	address.h
tid	TransactionIdentifier	address.h
exercise	ExerciseID	basic.h
Internal Variables		
Variable	Type	Where Typedef Declared
old_exercise	ExerciseID	basic.h
handle	int	Standard
Calls		
Function	Where Described	
network get net handle	Section 2.1.1.3.2.12.1	
network get exercise id	Section 2.1.1.3.1.16.1	
AssocUnsubscribe	See MCC CSCI Section 2.20.1.1.2	
network set exercise id	Section 2.1.1.3.1.49.1	
AssocSubscribe	See MCC CSCI Section 2.20.1.1.1	
veh set force	Section 2.6.10.6.2	
filter set force	Section 2.5.14.6.1	
filter set location	Section 2.5.14	
network use activation	Section 2.1.1.3.1.68.2	

Table 2.1-84: process_activate_request Information.

2.1.1.3.2.2 alert_status.c

(/simnet/release/src/vehicle/libsrc/libRcvNet/alert_status.c)

Code in this file is not used in the Version 6.6 release.

2.1.1.3.2.3 collision.c

(/simnet/release/src/vehicle/libsrc/libRcvNet/collision.c)

Includes:

```

"mass_std.h"
"sim_types.h"
"dgi_std.h"
"sim_cig_if.h"
"ctype.h"
"librva.h"
"rcv_loc.h"
"net_stats.h"
"pro_sim.h"
"pro_assoc.h"
"bigwheel.h"
"libhull.h"

```

Declarations: network_debug = FALSE

2.1.1.3.2.3.1 process_collision

This routine processes a Collision PDU. This routine makes a call to **network_send_collision_response()** to acknowledge receipt of the packet. A call is also made to **collision_check_veh_coll_at()**, located in libbigwh, to determine the direction from which the vehicle was hit.

Parameters		
Parameter	Type	Where Typedef Declared
p	register pointer to CollisionVariant	p_sim.h
originator	pointer to SimulationAddress	address.h
tid	TransactionIdentifier	address.h
Internal Variables		
Variable	Type	Where Typedef Declared
hash id	int	Standard
my_vehicle_id	pointer to VehicleID	basic.h
rva find hash entry()	extern int	Standard
Calls		
Function	Where Described	
rva find hash entry	Section 2.5.12.11.11	
network get vehicle id	Section 2.1.1.3.1.22.1	
network send collision response	Section 2.1.1.3.1.7.1	
collision check veh coll at	Section 2.5.10.6.1	

Table 2.1-85: process_collision Information.

2.1.1.3.2.4 deactivate.c

(./simnet/release/src/vehicle/libsrc/libRcvNet/deactivate.c)

Includes:

"ctype.h"
 "rcv_loc.h"
 "net_stats.h"
 "pro_sim.h"
 "pro_assoc.h"

2.1.1.3.2.4.1 process_deactivate_me

This routine processes a Deactivate PDU that is intended to deactivate the vehicle. The routine makes a call to libSendNet to acknowledge receipt with a Deactivate Response PDU and initiates the deactivation.

Parameters		
Parameter	Type	Where Typedef Declared
pkt	pointer to DeactivateRequestVariant	p_sim.h
originator	pointer to SimulationAddress	address.h
tid	TransactionIdentifier	address.h
Calls		
Function	Where Described	
NOTE FOR US		
network send deactivate response	Section 2.1.1.3.1.9.1	
beep	Section 2.6.4.1.1	
deactivate simulation	Section 2.5.1.1.4	

Table 2.1-86: process_deactivate_me Information.

2.1.1.3.2.4.2 process_deactivate_other

This routine processes a Deactivate PDU that informs the invoking process of another vehicle's deactivation. The routine makes a call to librva in order to cease dead reckoning and displaying that vehicle.

Parameters		
Parameter	Type	Where Typedef Declared
pkt	pointer to DeactivateRequestVariant	p_sim.h
Calls		
Function	Where Described	
rva forget about other vehicle	Section 2.5.12	

Table 2.1-87: process_deactivate_other Information.

2.1.1.3.2.5 fire.c

(/simnet/release/src/vehicle/libsrc/libRcvNet/fire.c)

Includes:

```

"rcv_loc.h"
"net_stats.h"
"pro_sim.h"
"mass_std.c.h"
"dgi_stdg.h"
"sim_cig_if.h"
"libkin.h"
"bigwheel.h"
"libhull.h"
"libimps.h"
"librva.h"
"libveh.h"

```

Declarations: remote_vehicles[]

2.1.1.3.2.5.1 process_fire

This routine processes a Fire PDU that notifies the invoking process that a shell or missile was fired. The routine determines whether it is necessary for the vehicle to display any effects (such as a muzzle flash) or make any sounds.

Parameters		
Parameter	Type	Where Typedef Declared
p	pointer to FireVariant	p_sim.h
Internal Variables		
Variable	Type	Where Typedef Declared
map_index	int	Standard
r 2	REAL	sim_types.h
vec	pointer to REAL	sim_types.h
cig_get_current_range_sqrd	extern REAL	sim_types.h
Calls		
Function	Where Described	
kinematics get o to h	Section 2.5.8.2.4	
kinematics range squared	Section 2.5.8.10.1	
cig_get_current_range_sqrd	Section 2.5.12.21.1	
map_get_ammo_entry_from_network_type	Section 2.6.11.2.1	
NOTE FOR US		
impacts queue effect	Section 2.5.15.1.3	

Table 2.1-88 process_fire Information.

2.1.1.3.2.6 fire_probe.c

(/simnet/release/src/vehicle/libsrc/libRcvNet/fire_probe.c)

Code in this file is not used in the Version 6.6 release.

2.1.1.3.2.7 idiot_check.c

(/simnet/release/src/vehicle/libsrc/libRcvNet/idiot_check.c)

Code in this file is not used in the Version 6.6 release.

2.1.1.3.2.8 impact.c

(/simnet/release/src/vehicle/libsrc/libRcvNet/impact.c)

Includes:

```

"rcv_loc.h"
"net_stats.h"
"pro_sim.h"
"pro_assoc.h"
"sim_macros.h"
"mass_stdc.h"
"dgi_stdg.h"
"sim_cig_if.h"
"libkin.h"
"libhull.h"
"librva.h"
"libfail.h"
"libimps.h"
"libsound.h"
"libveh.h"

```

Declarations:

```

remote_vehicles[]
impact_debug
null_world_coordinates

```

2.1.1.3.2.8.1 process_hit_me

This routine processes an Impact PDU that notifies the invoking process of a shell or missile impact with the vehicle. A call is made to libSendNet to acknowledge receipt of the packet, and a call is made to **veh_impact_me()** in order to assess damages.

Parameters		
Parameter	Type	Where Typedef Declared
p	pointer to ImpactVariant	p_sim.h
originator	pointer to SimulationAddress	address.h
tid	TransactionIdentifier	address.h
Calls		
Function	Where Described	
veh impact me	Section 2.1.1.3.2.8.3	
network send impact response	Section 2.1.1.3.1.26.1	

Table 2.1-89: process_hit_me Information.

2.1.1.3.2.8.2 process_hit_other

This routine processes an Impact PDU that notifies the invoking process of an impact with an object other than the vehicle itself. A determination is made as to whether the impact was with another vehicle or with the ground, or if it went out of the range of the terrain database. The appropriate processing routine is then called.

Parameters		
Parameter	Type	Where Typedef Declared
p	register pointer to ImpactVariant	p_sim.h
Calls		
Function	Where Described	
veh_impact_other	Section 2.1.1.3.2.8.4	
ground_impact	Section 2.1.1.3.2.8.5	
non_impact	Section 2.1.1.3.1.33	

Table 2.1-90: process_hit_other Information.

2.1.1.3.2.8.3 veh_impact_me

This routine assesses damages due to an impact with the vehicle. The ammo type and failure are determined, and the appropriate effects are shown.

Parameters		
Parameter	Type	Where Typedef Declared
p	register pointer to ImpactVariant	p_sim.h
Calls		
Function	Where Described	
map_get_ammo_entry_from_network_type	Section 2.6.11.2.1	
impacts_queue_effect	Section 2.5.15.1.3	
failure_check_cat_kill	Section 2.2.4.1.9 Section 2.3.4.1.3	

Table 2.1-91: veh_impact_me Information.

2.1.1.3.2.8.4 veh_impact_other

This routine determines which sounds to make and which visual effects to show when an impact is made with another vehicle.

Parameters		
Parameter	Type	Where Typedef Declared
p	register pointer to ImpactVariant	p_sim.h
Internal Variables		
Variable	Type	Where Typedef Declared
ammo_type	int	Standard
r 2	double	Standard
Calls		
Function	Where Described	
map_get_amm0_entry_from_network_type	Section 2.6.11.2.1	
kinematics_range_squared	Section 2.5.8.10.1	
impacts_queue_effect	Section 2.5.15.1.3	

Table 2.1-92: veh_impact_other Information.

2.1.1.3.2.8.5 ground_impact

This routine determines the sounds to make and the visual effects to show when an impact is made with the ground.

Parameters		
Parameter	Type	Where Typedef Declared
p	register pointer to ImpactVariant	p_sim.h
Internal Variables		
Variable	Type	Where Typedef Declared
ammo_type	int	Standard
r 2	double	Standard
Calls		
Function	Where Described	
map_get_amm0_entry_from_network_type	Section 2.6.11.2.1	
kinematics_range_squared	Section 2.5.8.10.1	
impacts_queue_effect	Section 2.5.15.1.3	

Table 2.1-93: ground_impact Information.

2.1.1.3.2.9 indir_fire.c

(/simnet/release/src/vehicle/libsrc/libRcvNet/indir_fire.c)

Includes:

```

"stdio.h"
"rcv_loc.h"
"net_stats.h"
"pro_sim.h"
"mun_type.h"
"obj_type.h"
"sim_macros.h"
"mass_std.h"
"dgi_std.h"
"sim_cig_if.h"
"libkin.h"
"bigwheel.h"
"libhull.h"
"librva.h"
"libfail.h"
"libimps.h"
"libcig.h"
"libsound.h"

```

Declarations:

```

got_diag_eff
ind_fire_debug

```

2.1.1.3.2.9.1 process_indirect_fire

This routine processes an Indirect Fire PDU that notifies the invoking process of the impacts due to indirect fire. The routine calculates the range of the impacts, assesses any damages, and determines whether it is necessary for the vehicle to display effects or make sounds.

Parameters		
Parameter	Type	Where Typedef Declared
p	pointer to IndirectFireVariant	p_sim.h
Internal Variables		
Variable	Type	Where Typedef Declared
i	register int	Standard
delay	register int	Standard
ammo_type	int	Standard
Calls		
Function	Where Described	
map_get_ammo_entry_from_network_type	Section 2.6.11.2.1	
cig_get_current_range_sqrd	Section 2.5.12.21.1	
kinematics_range_squared	Section 2.5.8.10.1	
cig_get_current_range_sqrd	Section 2.5.12.21.1	
failure_check_indir_fire_damages	Section 2.3.4.1.10 Section 2.3.4.1.4	
impacts_queue_effect	Section 2.5.15.1.3	

Table 2.1-94: process_indirect_fire Information.

2.1.1.3.2.10 laser_range.c

(./simnet/release/src/vehicle/libsrc/libRcvNet/laser_range.c)

Code in this file is a stub for files that do not recognize laser packets.

2.1.1.3.2.11 markers.c

(./simnet/release/src/vehicle/libsrc/libRcvNet/markers.c)

Includes:

```
"rcv_loc.h"
"net_stats.h"
"pro_sim.h"
"sim_macros.h"
"mass_std.h"
"dgi_stdg.h"
"sim_cig_if.h"
"libhull.h"
"libkin.h"
"librva.h"
```

2.1.1.3.2.11.1 process_markers

This routine processes a Marker PDU. Minefield markers are processed by librva as static vehicles.

Parameters		
Parameter	Type	Where Typedef Declared
<i>r</i>	register pointer to MarkerVariant	p_sim.h
Calls		
Function	Where Described	
NOTE FOR US		
rva_process_markers	Section 2.5.12.13.3	

Table 2.1-95: process_markers Information.

2.1.1.3.2.12 network_init.c

(.simnet/release/src/vehicle/libsrc/libRcvNet/network_init.c)

Includes:

```
"rcv_loc.h"
"itab.h"
"errno.h"
"net/network.h"
"pro_assoc.h"
"pro_sim.h"
"pro_num.h"
"assoc.h"
"p2p.h"
```

Defines:

```
NETWORK_DEVICE_NAME_SIZE
ASSOC_DEF_FILE
```

Declarations:

```
network_device_name[NETWORK_DEVICE_NAME_SIZE]
layer_to_use
assoc_net_handle
```

2.1.1.3.2.12.1 network_get_net_handle

The libnetif module supports multiple networks. This routine obtains a network handle, which is a low integer referring to the connection to a particular network.

Return Values		
Return Value	Type	Meaning
assoc net handle	int	the network handle

Table 2.1-96: network_get_net_handle Information.

2.1.1.3.2.12.2 network_set_net_layer

This routine sets up the network to use either the association or point to point layer.

Parameters		
Parameter	Type	Where Typedef Declared
layer	int	Standard
Return Values		
Return Value	Type	Meaning
0	int	indication of success

Table 2.1-97: network_set_net_layer Information.

2.1.1.3.2.12.3 network_init

This routine obtains the connection to the network using either the association or point to point layer and sets the parameters for the network. Timestamping is disabled and the clock is reset to zero. The routine also obtains the simulation address and determines the vehicle ID.

Internal Variables		
Variable	Type	Where Typedef Declared
sim_address	SimulationAddress	address.h
vid	VehicleID	basic.h
Errors		
Error	Reason for Error	
Error disabling timestamping	problems disabling timestamping	
Error initting net timer	problems resetting clock to zero	
Calls		
Function	Where Described	
set_process_pkt_fn	Section 2.1.1.3.2.18.1	
PointToPointOpen	Section 2.1.1.2.2.1.1	
AssocError	See MCC CSCI Section 2.20.1.10.1	
AssocOpen	See MCC CSCI Section 2.20.1.5.1	
net_stamp_disable	See MCC CSCI Section 2.20.2.12.2	
net_init_time	See MCC CSCI Section 2.20.2.8.4	
AssocSubscribe	See MCC CSCI Section 2.20.1.1.1	
assoc_net_handle		
AssocGetSimAddress	See MCC CSCI Section 2.20.1.9.1	
network_set_vehicle_id	Section 2.1.1.3.1.55.1	

Table 2.1-98: network_init Information.

2.1.1.3.2.12.4 network_set_network_device

This routine sets the network device name used for the network connection to *device*.

Parameters		
Parameter	Type	Where Typedef Declared
device	pointer to char	Standard

Table 2.1-99: network_set_network_device Information.

2.1.1.3.2.12.5 network_get_network_device

This routine returns the network device name for a network connection.

Return Values		
Return Value	Type	Meaning
network_device_name	pointer to char	the name of the network device

Table 2.1-100: network_get_network_device Information.

2.1.1.3.2.13 network_test.c

(./simnet/release/src/vehicle/libsrc/libRcvNet/network_test.c)

Code in this file is not compiled in the Version 6.6 release.

2.1.1.3.2.14 not_open_net.c

(./simnet/release/src/vehicle/libsrc/libRcvNet/not_open_net.c)

Includes: "rcv_loc.h"

2.1.1.3.2.14.1 network_dont_really_open_up_ethernet

This routine sets the variable, *using_ethernet*, to FALSE. This routine is used during debugging in order to run the simulator without connecting to the network.

2.1.1.3.2.15 open_net.c

(./simnet/release/src/vehicle/libsrc/libRcvNet/open_net.c)

Includes: "rcv_loc.h"

2.1.1.3.2.15.1 network_really_open_up_ethernet

This routine sets the variable, *using_ethernet*, to TRUE.

2.1.1.3.2.16 print_stats.c

(/simnet/release/src/vehicle/libsrc/libRcvNet/print_stats.c)

Includes:

```
"rcv_loc.h"
"net_stats.h"
"pro_sim.h"
"pro_data.h"
"pro_mgmt.h"
```

2.1.1.3.2.16.1 network_print_statistics

This routine prints statistics about the number and types of packets sent on the network. It prints the packets received, the packets for the invoking process, and the packets sent for the Simulation Packet types, Data Analysis Packet types and Management Packet types.

Calls	
Function	Where Described
timers get current tick	Section 2.6.3.1.1

Table 2.1-101: network_print_statistics Information.**2.1.1.3.2.17 print_vimp.c**

(/simnet/release/src/vehicle/libsrc/libRcvNet/print_vimp.c)

Code in this file is not compiled in the Version 6.6 release.

2.1.1.3.2.18 proc_a_pkt.c

(/simnet/release/src/vehicle/libsrc/libRcvNet/proc_a_pkt.c)

Includes:

```
"stdio.h"
"bbd.h"
"rcv_loc.h"
"net_stats.h"
"pro_sim.h"
"pro_mgmt.h"
"pro_data.h"
"pro_stlth.h"
"pro_assoc.h"
"assoc.h"
"veh_type.h"
"p_faad.h"
"p_num.h"
```

Declarations:

```
process_pkt_fn()
keyboard_reconstitute
```

This file contains routines which sort the packets received from the network and call the appropriate modules of vehicles code for processing. First the packet is assigned to a major category: either Management Datagram, Data Collection Datagram, Simulation

Datagram, Simulation Transaction, or Data Collection Transaction. Each of the major categories distributes the packet to a routine contained within libRcvNet which then passes the packet to the appropriate module for processing. For example, `process_a_packet()` assigns a Marker PDU to the routine `do_protocol_on_sim_packet()`, located in this file. `do_protocol_on_sim_packet()` calls the routine `process_markers()`, located in the libRcvNet file "markers.c". `process_markers()` then calls `rva_process_markers()`, located in librva, which processes the information from the Marker PDU.

2.1.1.3.2.18.1 `set_process_pkt_fn`

This routine sets the association layer process packet function to *fn*. The default is *AssocReceivePDU*.

Parameters		
Parameter	Type	Where Typedef Declared
<code>fn</code>	function which returns a pointer to int	Standard

Table 2.1-102: `set_process_pkt_fn` Information.

2.1.1.3.2.18.2 `do_protocol_on_mgmt_packet`

This routine is not used in the version 6.6 release.

2.1.1.3.2.18.3 `do_protocol_on_data_analysis_packet`

This routine processes the `statusQueryPDUKind` and the `laserRangePDUKind`; other data analysis datagrams are ignored.

Parameters		
Parameter	Type	Where Typedef Declared
<code>pkt</code>	pointer to <code>DataCollectionPDU</code>	<code>p_data.h</code>
<code>exercise</code>	<code>ExerciseID</code>	<code>basic.h</code>
Calls		
Function	Where Described	
NOTE RECEIVED		
<code>process status query</code>	Section 2.1.1.3.2.34.1	
<code>process laser range</code>	Section 2.1.1.3.2	

Table 2.1-103: `do_protocol_on_data_analysis_packet` Information.

2.1.1.3.2.18.4 do_protocol_on_sim_packet

This routine processes the:

```

deactivateRequestPDUKind
vehicleAppearancePDUKind
radiatePDUKind
firePDUKind
impactPDUKind
indirectFirePDUKind
serviceRequestPDUKind
resupplyOfferPDUKind
resupplyReceivedPDUKind
resupplyCancelPDUKind
markerPDUKind.

```

All other simulation datagrams are ignored.

Parameters		
Parameter	Type	Where Typedef Declared
pkt	pointer to SimulationPDU	p_sim.h
Calls		
Function	Where Described	
NOTE RECEIVED		
process deactivate other	Section 2.1.1.3.2.4.2	
process update	Section 2.1.1.3.2.37.1	
process radiate		
process fire	Section 2.1.1.3.2.5.1	
process hit other	Section 2.1.1.3.2.8.2	
process indirect fire	Section 2.1.1.3.2.9.1	
process service request	Section 2.1.1.3.2.32.1	
process resupply offer	Section 2.1.1.3.2.30.1	
process resupply received	Section 2.1.1.3.2.31.1	
process resupply cancel	Section 2.1.1.3.2.29.1	
process markers	Section 2.1.1.3.2.11.1	

Table 2.1-104: do_protocol_on_sim_packet Information.

2.1.1.3.2.18.5 process_sim_transaction

This routine processes the:

deactivateRequestPDUKind if the simulator is in simulation state
 activateRequestPDUKind
 collisionPDUKind if the simulator is in simulation state
 impactPDUKind if the simulator is in simulation state
 repairRequestPDUKind if the simulator is in simulation state.

All other simulation transactions are ignored.

Parameters		
Parameter	Type	Where Typedef Declared
pkt	pointer to SimulationPDU	p_sim.h
originator	pointer to SimulationAddress	address.h
transID	TransactionIdentifier	address.h
Calls		
Function	Where Described	
NOTE RECEIVED		
sim state simulating	Section 2.5.1.1.11	
process deactivate me	Section 2.1.1.3.2.4.1	
process activate request	Section 2.1.1.3.2.1.1	
process collision	Section 2.1.1.3.2.3.1	
process hit me	Section 2.1.1.3.2.8.1	
process repair	Section 2.1.1.3.2.28.1	

Table 2.1-105: process_sim_transaction Information.

2.1.1.3.2.18.6 process_dc_transaction

This routine processes the statusQueryPDUKind; all other data analysis transactions are ignored.

Parameters		
Parameter	Type	Where Typedef Declared
pkt	pointer to DataCollectionPDU	p_data.h
originator	pointer to SimulationAddress	address.h
tid	TransactionIdentifier	address.h
exercise	ExerciseID	basic.h
Calls		
Function	Where Described	
NOTE RECEIVED		
process_query_me	Section 2.1.1.3.2.34.2	

Table 2.1-106: process_dc_transaction Information.

2.1.1.3.2.18.7 reconstitute_from_keyboard

This routine sets the variable *keyboard_reconstitute* equal to 1, signalling *process_a_packet()* to act as if it just received an Activate Packet. It is used for debugging purposes.

2.1.1.3.2.18.8 process_a_packet

This routine assigns each packet to a category based upon its protocol number. Once the packet is assigned, the routine then calls the appropriate processing routines.

Internal Variables		
Variable	Type	Where Typedef Declared
data	pointer to char	Standard
len	int	Standard
group	MulticastGroupID	p_assoc.h
protocol	AssociationUserProtocol	p_assoc.h
primitive	long int	Standard
originator	SimulationAddress	address.h
respondent	SimulationAddress	address.h
transID	TransactionIdentifier	address.h
Return Values		
Return Value	Type	Meaning
TRUE	BOOLEAN	procedure was successful
FALSE	BOOLEAN	not using ethernet, do not have current protocol version, or the process packet function does not return 0
Calls		
Function	Where Described	
reconstitute_vehicle	Section 2.2.1.1.10 Section 2.3.1.1.12 Section 2.4.1.1.10	
network get net handle	Section 2.1.1.3.2.12.1	
process sim transaction	Section 2.1.1.3.2.18.5	
process_pkt fn	Section 2.1.1.3.2	
sim state simulating	Section 2.5.1.1.11	
do_protocol on sim_packet	Section 2.1.1.3.2.18.4	
process dc transaction	Section 2.1.1.3.2.18.6	
do_protocol on data analysis packet	Section 2.1.1.3.2.18.3	
do_protocol on mgmt_packet	Section 2.1.1.3.2.18.2	
do_protocol on stealth_packet	Section 2.1.1.3.2	
do_protocol on faad_packet	Section 2.1.1.3.2	

Table 2.1-107: process_a_packet Information.

2.1.1.3.2.19 **prot_faad.c** (./simnet/release/src/vehicle/libsrc/libRcvNet/prot_faad.c)

This file is a stub for files that do not recognize FAAD packets.

2.1.1.3.2.20 **prot_ivis.c** (./simnet/release/src/vehicle/libsrc/libRcvNet/prot_ivis.c)

This file is a stub for files that do not recognize IVIS packets.

2.1.1.3.2.21 **prot_laser.c** (./simnet/release/src/vehicle/libsrc/libRcvNet/prot_laser.c)

This file is a stub for files that do not recognize Laser Effects packets.

2.1.1.3.2.22 **prot_stealth.c** (./simnet/release/src/vehicle/libsrc/libRcvNet/prot_stealth.c)

This file is a stub for vehicles that do not recognize Stealth packets. It should be linked in by only these vehicles and should NEVER be called.

2.1.1.3.2.23 **radiate.c** (./simnet/release/src/vehicle/libsrc/libRcvNet/radiate.c)

This file is a stub for files that do not recognize Radiate packets.

2.1.1.3.2.24 **rad_state.c** (./simnet/release/src/vehicle/libsrc/libRcvNet/rad_state.c)

Includes:

```
"rcv_loc.h"
"net_stats.h"
"libnetwork.h"
"pro_sim.h"
"pro_assoc.h"
"p_num.h"
"p_faad.h"
```

2.1.1.3.2.24.1 **process_radiating_state**

This routine processes Radiating State PDUs using the datagram service.

Parameters		
Parameter	Type	Where Typedef Declared
p	register pointer to RadiatingStateVariant	
Calls		
Function	Where Described	
NOTE FOR US		
controls radar state	Section 2.2.2	

Table 2.1-108: process_radiating_state Information.

2.1.1.3.2.25 **rcv_loc.c** (./simnet/release/src/vehicle/libsrc/libRcvNet/rcv_loc.c)

Includes:

"rcv_loc.h"
"net_stats.h"

Declarations:

using_ethernet = TRUE
net_handle
packets_received[maxProtocolFamily+1][maxPacketsPerFamily]
packets_for_us[maxProtocolFamily+1][maxPacketsPerFamily]

2.1.1.3.2.26 **rcv_loc.h** (./simnet/release/src/vehicle/libsrc/libRcvNet/rcv_loc.h)

2.1.1.3.2.27 **really.c** (./simnet/release/src/vehicle/libsrc/libRcvNet/really.c)

Includes: "rcv_loc.h"

2.1.1.3.2.27.1 **network_can_i_really_use_network**

This routine returns the value of the *using_ethernet* flag, which determines whether the network can be used.

Return Values		
Return Value	Type	Meaning
using_ethernet	int	if <i>using_ethernet</i> is TRUE, the network can be used; if <i>using_ethernet</i> is FALSE, the network cannot be used

Table 2.1-109: **network_can_i_really_use_network** Information.

2.1.1.3.2.28 repair.c (./simnet/release/src/vehicle/libsrc/libRcvNet/repair.c)

Includes:

"rcv_loc.h"
 "net_stats.h"
 "libnetwork.h"
 "pro_assoc.h"

2.1.1.3.2.28.1 process_repair

This routine processes a Repair Request PDU.

Parameters		
Parameter	Type	Where Typedef Declared
p	pointer to RepairRequestVariant	p_sim.h
originator	pointer to SimulationAddress	address.h
tid	TransactionIdentifier	address.h
Calls		
Function	Where Described	
NOTE FOR US		
repair request	Section 2.3.4.2.1	

Table 2.1-110: process_repair Information.

2.1.1.3.2.29 resupp_canc.c

(./simnet/release/src/vehicle/libsrc/libRcvNet/resupp_canc.c)

Includes:

"rcv_loc.h"
 "net_stats.h"
 "pro_sim.h"

2.1.1.3.2.29.1 process_resupply_cancel

This routine processes Resupply Cancel PDUs.

Parameters		
Parameter	Type	Where Typedef Declared
p	pointer to ResupplyCancelVariant	p_sim.h
Internal Variables		
Variable	Type	Where Typedef Declared
my_id	pointer to VehicleID	basic.h
Calls		
Function	Where Described	
network_get_vehicle_id	Section 2.1.1.3.1.22.1	
VEHICLE_IDS_EQUAL	sim_macros.h	
NOTE FOR US		
resupply_offer_canceled	Section 2.2.5.3.32	
resupply_request_canceled	Section 2.3.5.3.33	

Table 2.1-111: process_resupply_cancel Information.

2.1.1.3.2.30 resupp_offer.c

(/simnet/release/src/vehicle/libsrc/libRcvNet/resupp_offer.c)

Includes:

"rcv_loc.h"

"net_stats.h"

"pro_sim.h"

2.1.1.3.2.30.1 process_resupply_offer

This routine processes Resupply Offer PDUs.

Parameters		
Parameter	Type	Where Typedef Declared
p	pointer to ResupplyVariant	p_sim.h
Internal Variables		
Variable	Type	Where Typedef Declared
my_id	pointer to VehicleID	basic.h
Calls		
Function	Where Described	
network get vehicle id	Section 2.1.1.3.1.22.1	
VEHICLE IDS EQUAL	sim macros.h	
NOTE FOR US		
resupply offer packet	Section 2.2.5.3.12	

Table 2.1-112: process_resupply_offer Information.

2.1.1.3.2.31 resupp_recvd.c

(./simnet/release/src/vehicle/libsrc/libRcvNet/resupp_recvd.c)

Includes:

"rcv_loc.h"
 "net_stats.h"
 "pro_sim.h"

2.1.1.3.2.31.1 process_resupply_received

This routine processes Resupply Received PDUs.

Parameters		
Parameter	Type	Where Typedef Declared
p	pointer to ResupplyVariant	p_sim.h
Internal Variables		
Variable	Type	Where Typedef Declared
my_id	pointer to VehicleID	basic.h
Calls		
Function	Where Described	
network_get_vehicle_id	Section 2.1.1.3.1.22.1	
VEHICLE_IDS_EQUAL	sim_macros.h	
NOTE_FOR_US		
resupply_thank_you_packet	Section 2.2.5.3.13	

Table 2.1-113: process_resupply_received Information.

2.1.1.3.2.32 service_req.c

(/simnet/release/src/vehicle/libsrc/libRcvNet/service_req.c)

Includes:

"rcv_loc.h"
 "net_stats.h"
 "pro_sim.h"

2.1.1.3.2.32.1 process_service_request

This routine processes Service Request PDUs.

Parameters		
Parameter	Type	Where Typedef Declared
p	pointer to ResupplyVariant	p_sim.h
Internal Variables		
Variable	Type	Where Typedef Declared
my_id	pointer to VehicleID	basic.h
Calls		
Function	Where Described	
network get vehicle id	Section 2.1.1.3.1.22.1	
VEHICLE IDS EQUAL	sim macros.h	
NOTE FOR US		
resupply feed me packet	Section 2.2.5.3.14	

Table 2.1-114: process_service_request Information.**2.1.1.3.2.33 show_effect.c**

(/simnet7/release/src/vehicle/libsrc/libRcvNet/show_effect.c)

This file is not used in the Version 6.6 release.

2.1.1.3.2.34 status_query.c

(./simnet7/release/src/vehicle/libsrc/libRcvNet/status_query.c)

Includes:

"rcv_loc.h"
 "net_stats.h"
 "pro_data.h"
 "pro_assoc.h"
 "libnetwork.h"

2.1.1.3.2.34.1 process_status_query

This routine processes Status Query PDUs by calling libSendNet to respond with the appropriate Status PDU using the datagram service.

Parameters		
Parameter	Type	Where Typedef Declared
pkt	pointer to StatusQueryVariant	p_data.h
exercise	ExerciseID	basic.h
Calls		
Function	Where Described	
network respond to query pkt	Section 2.1.1.3.1.63.2	

Table 2.1-115: process_status_query Information.

2.1.1.3.2.34.2 process_query_me

This routine processes Status Query PDUs by calling libSendNet to send a transaction response to the query.

Parameters		
Parameter	Type	Where Typedef Declared
pkt	pointer to StatusQueryVariant	p_data.h
originator	pointer to SimulationAddress	address.h
tid	TransactionIdentifier	address.h
exercise	ExerciseID	basic.h
Calls		
Function	Where Described	
NOTE FOR US		
network respond to query trans	Section 2.1.1.3.1.63.1	

Table 2.1-116: process_query_me Information.

2.1.1.3.2.35 tgt_fire_cmd.c

(./simnet7/release/src/vehicle/libsrc/libRcvNet/tgt_fire_cmd.c)

The code in this file is not used in the Version 6.6 release.

2.1.1.3.2.36 tgt_handoff.c

(./simnet/release/src/vehicle/libsrc/libRcvNet/tgt_handoff.c)

The code in this file is not used in the Version 6.6 release.

2.1.1.3.2.37 tgt_vis.c

(./simnet/release/src/vehicle/libsrc/libRcvNet/tgt_vis.c)

The code in this file is not used in the Version 6.6 release.

2.1.1.3.2.38 veh_appear.c

(./simnet/release/src/vehicle/libsrc/libRcvNet/veh_appear.c)

Includes:

"rcv_loc.h"
 "net_stats.h"
 "pro_sim.h"
 "sim_macros.h"
 "mass_std.h"
 "dgi_stdg.h"
 "sim_cig_if.h"
 "libhull.h"
 "libkin.h"
 "librva.h"

2.1.1.3.2.38.1 process_update

This routine processes the Vehicle Appearance PDUs by calling the librva routine, **rva_process_update()**.

Parameters		
Parameter	Type	Where Typedef Declared
p	register pointer to VehicleAppearanceVariant	p_sim.h
Calls		
Function	Where Described	
NOTE FOR US		
rva_process_update	Section 2.5.12.20.5	

Table 2.1-117: process_update Information.

2.1.2 CIG Interface Software

The CIG (Computer Image Generation) System is accessed through a set of C software libraries organized in a manner similar to that of the network interface modules. These libraries provide the high level interface needed to easily communicate with the CIG. The files included in this CSC are shown in Figure 2.1-3. The third level CSCs associated with the CIG Interface software are CIG Device Interface and CIG/SIM Buffer Interface.

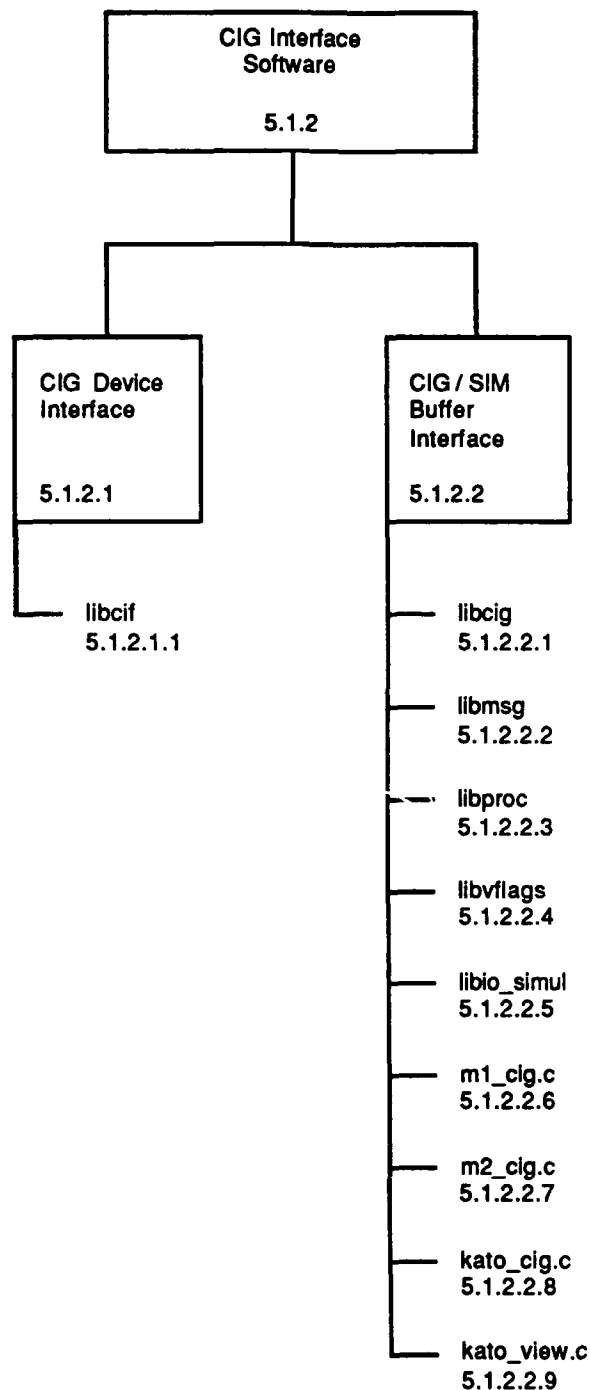


Figure 2.1-3: CIG Interface software structure.

2.1.2.1 CIG Device Interface

The CIG Device Interface provides high level device independent routines to communicate with the CIG. The M1, M2, and Stealth simulators use different Sim host computers, as well as different interface devices to communicate between the CIG and Sim host computers. The M1 communicates via a DR11W interface while the M2 communicates via a VME interface. The Stealth 3 communicates with a DR11W interface. The CIG Device Interface deals with this difference exclusively, thus allowing the actual interface device to be transparent to the user. It contains one CSU, libcif.

2.1.2.1.1 libcif

(./simnet/release/src/libsrc/libcif [libcif])

Libcif contains routines which allow the user to initialize the devices, and to obtain a piece of shared memory to use as the buffer to transfer data between the CIG and simulation host, and routines to start the transfer as well as those which indicate that the transfer has been completed.

2.1.2.1.1.1 connect.c

(./simnet/release/src/libsrc/libcif/connect.c)

This file contains a routine which connects the process to an interface.

Includes:

libcif.h

Defines:

DEBUG

2.1.2.1.1.1.1 cif_connect

This routine attempts to connect to the interface specified by the *his_interface* argument. This routine will wait until it connects to its peer if the *wait* flag is TRUE; otherwise, it will return immediately with an indication of success or failure. A pointer to the send buffer is returned in *send_buf*; the size of the send buffer is returned in *send_buf_size*; and the size of the receive buffer is returned in *receive_buf_size*.

Under Chrysalis: A connection to a DR11 is attempted if the environment string "DR11_n address" is found in the environment, where *n* is the *his_interface* argument and *address* is the physical address that the DR11 is to use. The library looks for an environment variable *CIF_BUFFERS_n_TO_m* to determine the size of the buffers to use. *n* is the id of *my_interface* and *m* is the id of *his_interface*. The format of the environment string is "CIF_BUFFERS_n_TO_m read_buf_size write_buf_size". For the Butterfly, *his_interface* must be the unique integer associated with the peer process.

Under UNIX: For the DR11, *his_interface* corresponds to the minor device number of the DR11 device, i.e., 0 for /dev/dr0, 1 for /dev/dr1, etc.

Parameters		
Parameter	Type	Where Typedef Declared
his interface	unsigned	Standard
send buf	pointer to a pointer to char	Standard
send buf size	int	Standard
receive buf size	int	Standard
wait	int	Standard

Table 2.1-118: cif_connect Information.

If a Butterfly machine is used:

Internal Variables		
Internal Variable	Type	Where Typedef Declared
ptable	pointer to cif connection table	Section 2.1.2.1.1.5 libcif.h
name[80]	char	Standard
penv	pointer to char	Standard
pdr11	pointer to device	drdev.h
Return Values		
Return Value	Type	Meaning
-1	int	failed
0	int	successful
Errors		
Error Name	Reason for Error	
ENETDOWN	cif interface not initialized	
ENXIO	bad interface number	
EISCONN	is already connected	
EWOULDBLOCK	operation not started	
ENOBUFS	incompatible buffer size	
Calls		
Function	Where Described	
map_mregion	Standard library function for the Butterfly	
Make_Obj	Standard library function for the Butterfly	
map to multibus	Standard library function for the Butterfly	
Map_Obj	Standard library function for the Butterfly	
Name Bind	Standard library function for the Butterfly	
Find Value	Standard library function for the Butterfly	

Table 2.1-119: cif_connect Information for the Butterfly machine.

If a Masscomp machine is used:

Internal Variables		
Internal Variable	Type	Where Typedef Declared
ptable	pointer to cif_connection table	Section 2.1.2.1.1.5 libcif.h
pdr11	pointer to device	drdev.h
Return Values		
Return Value	Type	Meaning
-1	int	failed
0	int	successful
Errors		
Error Name	Reason for Error	
ENETDOWN	cif interface not initialized	
ENXIO	bad interface number	
EISCONN	is already connected	

Table 2.1-120: cif_connect Information for the Masscomp machine.

2.1.2.1.1.2 data.c

(./simnet/release/src/libsrc/libcif/data.c)

This file contains variable declarations and constant definitions to be used within libcif.

2.1.2.1.1.3 disconnect.c

(./simnet/release/src/libsrc/libcif/disconnect.c)

This file contains the routine which closes the connection with an interface.

Includes:

"libcif.h"

Defines:

SEBUG

2.1.2.1.1.3.1 cif_disconnect

This routine attempts to disconnect from the interface specified by the *his_interface* argument. This routine will wait until it is able to disconnect with its peer if the *wait* argument is TRUE, and will return with an indication of success if the connection has been broken. An indication of failure is returned if the connection cannot be broken.

For the DR11, *his_interface* corresponds to the minor device number of the DR11 device (i.e., 0 for /dev/dr0, 1 for /dev/dr1, etc.). For the Butterfly, *his_interface* must be the unique integer associated with the peer process.

Parameters		
Parameter	Type	Where Typedef Declared
his interface	int	Standard
wait	int	Standard

Table 2.1-121: cif_disconnect Information.

If a Butterfly machine is used:

Internal Variables		
Internal Variable	Type	Where Typedef Declared
ptable	pointer to cif connection table	Section 2.1.2.1.1.5 libcif.h
Return Values		
Return Value	Type	Meaning
-1	int	failure
0	int	success
Errors		
Error Name	Reason for Error	
ENETDOWN	communication interface not initialized	
ENOTCONN	not connected	
ENXIO	bad interface number	
Calls		
Function	Where Described	
unmap_mregion	Standard library function for the Butterfly	
unmap_to_multibus	Standard library function for the Butterfly	
Unmap_Obj	Standard library function for the Butterfly	
Name_Unbind	Standard library function for the Butterfly	

Table 2.1-122: cif_disconnect Information for the Butterfly machine.

If a Masscomp machine is used:

Internal Variables		
Internal Variable	Type	Where Typedef Declared
ptable	pointer to cif_connection_table	Section 2.1.2.1.1.5 libcif.h
Return Values		
Return Value	Type	Meaning
0	int	success
-1	int	failure
Errors		
Error Name	Reason for Error	
ENETDOWN	communication interface not initialized	
ENOTCONN	not connected	
ENXIO	bad interface number	

Table 2.1-123: cif_disconnect Information for the Masscomp machine.

2.1.2.1.1.4 init.c (./simnet/release/src/libsrc/libcif/init.c)

This file contains a routine which initializes the CIG interface driver.

Includes:
"libcif.h"

Defines:
DEBUG

2.1.2.1.1.4.1 cif_init

This routine initializes the communications driver. This routine takes only one argument, *interface*, which is the interface number for the caller. This routine is called only once to initialize this module.

For the DR11, the *interface* argument must be 0. For the Butterfly, the *interface* argument must be an integer in the range 0 to MAX_CIF_CONNECTIONS-1, and it must be unique for each communication process on the machine.

This call returns 0 if successful and -1 if not successful.

Parameters		
Parameter	Type	Where Typedef Declared
interface	int	Standard

Table 2.1-124: cif_init Information.

If a Butterfly machine is used:

Return Values		
Return Value	Type	Meaning
0	int	successful
-1	int	failed
Errors		
Error Name	Reason for Error	
EALREADY	interface already initialized	
ENXIO	bad interface number	

Table 2.1-125: cif_init Information for the Butterfly Machine.

If a Masscomp machine is used:

Internal Variables		
Internal Variable	Type	Where Typedef Declared
sbrk()	pointer to char	Standard
shmat()	pointer to char	Standard
cp	pointer to char	Standard
size	int	Standard
id	int	Standard
ptable	pointer to cif connection table	Section 2.1.2.1.1.5 libcif.h
dr11_page_paddr	long	Standard
i	int	Standard
buffers_vaddr	pointer to char	Standard
buffers_paddr	pointer to char	Standard
dr11_physical_addresses	pointer to int	Standard
f	pointer to FILE	Standard
total	int	Standard
Return Values		
Return Value	Type	Meaning
0	int	success
-1	int	failure
Errors		
Error Name	Reason for Error	
EALREADY	interface already initialized	
ENXIO	invalid interface number	
Calls		
Function	Where Described	
parce cif definition	Section 2.1.2.1.1.6.1	

Table 2.1-126: cif_init Information for the Masscomp Machine.

2.1.2.1.1.5 libcif.h

(./simnet/release/src/libsrc/libcif/libcif.h)

This file contains structure and constant definitions, as well as variable and function declarations, to be used in libcif.

2.1.2.1.1.6 parse.c

(./simnet/release/src/libsrc/libcif/parse.c)

This file contains the code to parse a definition file.

includes:

stdio.h

2.1.2.1.1.6.1 parse_cif_definition

This routine parses information in the file *filename* for the interface specified by the argument *n*. *readp* indicates where to return the read size. *writep* indicates where to return the write size.

The parameter file which is supplied is opened. The first character on a line is examined. Blank lines are skipped. Success is indicated if this line is the same as the requested line. Failure is indicated if the file is unopenable, or if the line is not found. This routine returns 0 if successful and -1 if not successful.

Parameters		
Parameter	Type	Where Typedef Declared
filename	pointer to char	Standard
n	int	Standard
readp	pointer to int	Standard
writep	pointer to int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
f	pointer to FILE	Standard
i	int	Standard
valid_linecount	int	Standard
str[100]	char	Standard
errbuf[80]	char	Standard
Return Values		
Return Value	Type	Meaning
-1	int	failed
0	int	successful

Table 2.1-127: parse_cif_definition Information.

2.1.2.1.1.7 receive.c

(/simnet/release/src/libsrc/libcif/receive.c)

This file contains the code which is used to receive a buffer.

Includes:

"libcif.h"

"rtc.h"

2.1.2.1.1.7.1 cif_receive

This routine receives a buffer from the peer specified by the *his_interface* assignment. A pointer to the buffer is returned in *pbuf*. *size* is the size of the buffer. If *wait* is true, this call will wait until the transfer is completed; otherwise, this call will return immediately with an indication of success or failure.

For the DR11, *his_interface* corresponds to the minor device number of the DR11 device (i.e., 0 for /dev/dr0, 1 for /dev/dr1, etc.). For the Butterfly, *his_interface* must be the unique integer associated with the peer process.

This routine returns 0 if successful and -1 if not successful.

Parameters		
Parameter	Type	Where Typedef Declared
<i>his_interface</i>	int	Standard
<i>pbuf</i>	char	Standard
<i>size</i>	int	Standard
<i>wait</i>	int	Standard

Table 2.1-128: *cif_receive* Information.

If a Butterfly machine is used:

Internal Variables		
Internal Variable	Type	Where Typedef Declared
ret	int	Standard
whichbuf	int	Standard
ptable	pointer to cif connection table	Section 2.1.2.1.1.5 libcif.h
Return Values		
Return Value	Type	Meaning
-1	int	failed
0	int	successful
Errors		
Error Name	Reason for Error	
ENETDOWN	cif interface not initialized	
ENXIO	bad interface number	
ENOTCONN	not connected	
EWouldBLOCK	operation not started	
Calls		
Function	Where Described	
dr11_receive	Section 2.1.2.1.1.7.2	
cif_disconnect	Section 2.1.2.1.1.3.1	

Table 2.1-129: cif_receive Information for Butterfly machine.

If a Masscomp machine is used:

Internal Variables		
Internal Variable	Type	Where Typedef Declared
ptable	pointer to struct cif connection table	Section 2.1.2.1.1.5 libcif.h
Return Values		
Return Value	Type	Meaning
-1	int	failed
0	int	successful
Errors		
Error Name	Reason for Error	
ENETDOWN	cif interface not initialized	
ENXIO	bad interface number	
ENOTCONN	not connected	
Calls		
Function	Where Described	
dr11_receive	Section 2.1.2.1.1.7.2	

Table 2.1-130: cif_receive Information for the Masscomp machine.

2.1.2.1.1.7.2 dr11_receive

This routine is called to receive a buffer via a DR11 interface card. *ptable* is a pointer into the *cif_connection_table*. *pbuf* is a pointer to the buffer to be received. *size* is the size of the buffer. *wait* is a flag which indicates if the buffer has been received.

Parameters		
Parameter	Type	Where Typedef Declared
ptable	pointer to cif connection table	Section 2.1.2.1.1.5 libcif.h
pbuf	pointer to pointer to char	Standard
size	int	Standard
wait	int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
ret	int	Standard
i	int	Standard
tmp	unsigned short	Standard
pdr11	pointer to register device	drdev.h
Return Values		
Return Value	Type	Meaning
-1	int	failed
0	int	successful
Errors		
Error Name	Reason for Error	
EINPROGRESS	operation started but not complete	
Calls		
Function	Where Described	
BSWAP	Section 2.1.2.1.1.5 libcif.h (macro definition)	
HIWORD	Section 2.1.2.1.1.5 libcif.h (macro definition)	
LOWORD	Section 2.1.2.1.1.5 libcif.h (macro definition)	

Table 2.1-131: dr11_receive Information.

2.1.2.1.1.8 send.c (./simnet/release/src/libsrc/libcif/send.c)

This file contains the code which is used to send a buffer to the CIG.

Includes:
"libcif.h"

2.1.2.1.1.8.1 cif_send

This routine sends the buffer *buf* of length *size* bytes to the peer specified by the *his_interface* assignment. A pointer to the buffer is returned in *pbuf*. If *wait* is true, this call will wait until the transfer is completed; otherwise, this call will return immediately with an indication of success or failure.

For the DR11, *his_interface* corresponds to the minor device number of the DR11 device (i.e., 0 for /dev/dr0, 1 for /dev/dr1, etc.). For the Butterfly, *his_interface* must be the unique integer associated with the peer process.

This routine returns 0 if successful and -1 if not successful.

Parameters		
Parameter	Type	Where Typedef Declared
his_interface	int	Standard
buf	pointer to char	Standard
size	int	Standard
wait	int	Standard

Table 2.1-132: cif_send Information.

If a Butterfly machine is used:

Internal Variables		
Internal Variable	Type	Where Typedef Declared
ret	int	Standard
whichbuf	int	Standard
ptable	pointer to cif_connection_table	Section 2.1.2.1.1.5 libcif.h
Return Values		
Return Value	Type	Meaning
-1	int	failed
0	int	successful
Errors		
Error Name	Reason for Error	
ENETDOWN	cif interface not initialized	
ENXIO	bad interface number	
ENOTCONN	not connected	
EWouldBLOCK	operation not started	
Calls		
Function	Where Described	
dr11_send	Section 2.1.2.1.1.8.2	
cif_disconnect	Section 2.1.2.1.1.3.1	

Table 2.1-133: cif_send Information for the Butterfly machine.

If a Masscomp machine is used:

Internal Variables		
Internal Variable	Type	Where Typedef Declared
ptable	pointer to cif connection table	Section 2.1.2.1.1.5 libcif.h
Return Values		
Return Value	Type	Meaning
-1	int	failed
0	int	successful
Errors		
Error Name	Reason for Error	
ENETDOWN	cif interface not initialized	
ENXIO	bad interface number	
ENOTCONN	not connected	
Calls		
Function	Where Described	
dr11 send	Section 2.1.2.1.1.8.2	

Table 2.1-134: cif_send Information for the Masscomp machine.

2.1.2.1.1.8.2 dr11_send

This routine is called to send a buffer via a DR11 interface card. *ptable* is a pointer into the *cif_connection_table*. *pbuf* is a pointer to the buffer to be received. *size* is the size of the buffer. *wait* is a flag which indicates if the buffer has been received.

Parameters		
Parameter	Type	Where Typedef Declared
ptable	pointer to cif_connection_table	Section 2.1.2.1.1.5 libcif.h
buf	pointer to char	Standard
size	int	Standard
wait	int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
ret	int	Standard
i	int	Standard
buf_paddr	pointer to char	Standard
pdr11	pointer to register device	drdev.h
Return Values		
Return Value	Type	Meaning
-1	int	failed
0	int	successful
Errors		
Error Name	Reason for Error	
EWOULDBLOCK	operation not started	
EINPROGRESS	operation started but not complete	
Calls		
Function	Where Described	
BSWAP	Section 2.1.2.1.1.5 libcif.h (macro definition)	
HIWORD	Section 2.1.2.1.1.5 libcif.h (macro definition)	
LOWORD	Section 2.1.2.1.1.5 libcif.h (macro definition)	

Table 2.1-135: dr11_send Information.

2.1.2.1.1.9 **uninit.c** (./simnet/release/src/libsrc/libcif/uninit.c)

This file contains a routine which uninitializes the communications driver.

2.1.2.1.1.9.1 **cif_uninit**

This routine uninitializes the communications driver and deallocates all resources allocated by the communications driver. This routine returns 0 if successful and -1 if not successful.

Internal Variables		
Internal Variable	Type	Where Typedef Declared
i	int	Standard
ret (Masscomp only)	int	Standard
ptable (Masscomp only)	pointer to cif_connection_table	Section 2.1.2.1.1.5 libcif.h
Return Values		
Return Value	Type	Meaning
0	int	success
-1	int	failure
Errors		
Error Name	Reason for Error	
ENETDOWN	communications interface is down	
Calls		
Function	Where Described	
cif_disconnect	Section 2.1.2.1.1.3.1	

Table 2.1-136: **cif_uninit** Information.

2.1.2.2 CIG-SIM Buffer Interface

The CIG-SIM interface consists of collection of messages which are used to transfer information between the CIG and the Simulation Host. These messages are placed into a buffer and transferred via the interface device. Messages sent from the CIG to the Simulation Host include informational messages such as those about local terrain, laser range, and hits, as well as setup messages that indicate which terrain database and DED are being used. Messages sent from the Simulation Host to the CIG contain information about the vehicle's current position and orientation, position, orientation and type for other visible vehicles in the world, and information necessary to paint ballistic effects such as tracers, muzzles, flashes, and armor and ground bursts. During setup, the Simulation Host also sends buffers containing the configuration of the viewports and ballistics trajectory information. (Reference Table 1.1-2 and *BBN Document #8912 - BBN GT100™ CIG to Simulation Host Interface Manual*.)

The CIG/SIM Buffer Interface consists of the following CSUs:

- libcig
- libmsg
- libproc
- libvflags
- libio_simul
- ml_cig.c
- m2_cig.c
- kato_cig.c
- kato_view.c

2.1.2.2.1 libcig (./simnet/release/src/vehicle/libsrc/libcig [libcig])

Libcig contains the routines to initialize, setup, and synchronize the CIG as well as actually send and receive the transfer buffers. The routines to send and receive buffers, `cig_send_buffer()` and `cig_receive_buffer()` respectively, are called every frame by `io_simul()`. Libcig contains the interfaces between libmsg and libcif and between libcif and libproc. Libcig sends formatted message buffers from libmsg to libcif. Libcig also receives raw buffers from libcif and sends them for processing to libproc.

2.1.2.2.1.1 check_sizes.c (./simnet/release/src/vehicle/libsrc/libcig/check_sizes.c)

Includes:

- "stdio.h"
- "sim_dfns.h"
- "mass_stdh.h"
- "dgi_stdg.h"
- "sim_cig_if.h"
- "cig_local.h"
- "libmsg.h"
- "gbuffer.h"

2.1.2.2.1.1.1 check_buffer_sizes

This routine checks that the buffer sizes being used are compatible with the transfer functionality. The parameter *num* indicates the buffer number to use. This routine checks that the buffer to be used is compatible with the program being run. Note that although the code is written to support more than one CIG, the PMTRADE system will contain only one CIG, CIG1.

Parameters		
Parameter	Type	Where Typedef Declared
num	int	Standard

Table 2.1-137: check_buffer_sizes Information.

2.1.2.2.1.1.2 cig_local.c

(./simnet/release/src/vehicle/libsrc/libcig/cig_local.c)

Includes:

"stdio.h"
 "sim_types.h"
 "sim_dfns.h"

This file contains default settings for the CIG system, and global variables used by the library.

2.1.2.2.1.1.3 cig_local.h

(./simnet/release/src/vehicle/libsrc/libcig/cig_local.h)

Includes "sim_types.h"

Global Variable Declarations:

using_graphics
 debug
 db_ptr
 db_override
 database_name[30]
 printbuffers
 send_buf[]
 receive_buf[]
 cig1_present
 cig2_present
 my_num
 req_send_size
 req_receive_size
 initial_send_size
 initial_receive_size
 initial_transfer_size
 use_requested_sizes

External Procedure Declaration:

check_buffer_sizes()**Defines:**

DR_THRU_KERNAL
 DR_MASSCOMP
 DR_DELTA
 BUF_SIZE

2.1.2.2.1.4 cig_no_op.c

(./simnet/release/src/vehicle/libsrc/libcig/cig_no_op.c)

Includes:

"sim_types.h"
 "stdio.h"
 "mass_stdc.h"
 "dgi_stdg.h"
 "sim_cig_if.h"
 "libmsg.h"

2.1.2.2.1.4.1 cig_prepare_no_op

This routine flushes all of the pointers in the buffer, and **push_msg_cig_ctl** (C_STOP) puts a message in the buffer which instructs the CIG to go into an idle state. The routine **flush_buffer()** clears the buffer of everything but other vehicle information; however, **cig_prepare_no_op()** is not usually called when other vehicle information is in the buffer. Functionally, the entire buffer is cleared by this call.

Calls	
Function	Where Described
flush buffer	Section 2.1.2.2.28.1
push msg cig ctl	Section 2.1.2.2.68.1

Table 2.1-138: cig_prepare_no_op Information.

2.1.2.2.1.5 cig_nuse_gra.c

(./simnet/release/src/vehicle/libsrc/libcig/cig_nuse_gra.c)

Includes:

"stdio.h"
 "sim_dfns.h"
 "mass_stdc.h"
 "dgi_stdg.h"
 "sim_cig_if.h"
 "cig_local.h"

2.1.2.2.1.5.1 cig_not_using_graphics

This routine sets the *using_graphics* flag to FALSE. This routine is used for testing purposes to notify the system to not issue any graphics commands.

2.1.2.2.1.6 **cig_prepare.c** (./simnet/release/src/vehicle/libsrc/libcig/cig_prepare.c)

```
"stdio.h"
  "errno.h"
  "sim_dfns.h"
  "mass_std.h"
  "dgi_stdg.h"
  "sim_cig_if.h"
  "cig_local.h"
```

2.1.2.2.1.6.1 **cig_prepare**

This routine prepares the CIG by setting up the interface between libcig and libcif. The parameter *ok_to_print* places the system into verbose mode, where status messages will be printed during operation.

Parameters		
Parameter	Type	Where Typedef Declared
ok to print	int	Standard
Errors		
Error	Reason for Error	
cif init failed	cif initialization failed	
Calls		
Function	Where Described	
cif init	Section 2.1.2.1.1.4.1	
cif connect	Section 2.1.2.1.1.1.1	
check buffer sizes	Section 2.1.2.2.1.1.1	
setup buffer ptrs	Section 2.1.2.2.1.32.1	
cig not using graphics (Butterfly Machine)	Section 2.1.2.2.1.5.1	

Table 2.1-139: **cig_prepare** Information.

2.1.2.2.1.7 cig_proc_buf.c

(./simnet/release/src/vehicle/libsrc/libcig/cig_proc_buf.c)

Includes:

```
"sim_types.h"
"stdio.h"
"mass_stdh.h"
"dgi_stdh.h"
"sim_cig_if.h"
"libcig.h"
"cig_local.h"
```

2.1.2.2.1.7.1 cig_process_buffer

This routine determines if a buffer is to be processed. If the buffer is not to be processed, i.e., `cig_not_ok_to_process_buffer()` is set, this routine will not allow libproc to receive the message. If the buffer is to be processed, a call is made to `process_buffer()` in libproc.

Calls	
Function	Where Described
<code>cig_not_ok_to_process_buffer</code>	Section 2.1.2.2.1.24.1
<code>process_buffer</code>	Section 2.1.2.2.3.8.1

Table 2.1-140: cig_process_buffer Information.

2.1.2.2.1.8 cig_r_start.c

(./simnet/release/src/vehicle/libsrc/libcig/cig_r_start.c)

Includes:

```
"sim_types.h"
"stdio.h"
"mass_stdh.h"
"dgi_stdh.h"
"sim_cig_if.h"
"libmsg.h"
"libcig.h"
"cig_local.h"
```

Global Variable Declarations:

```
print_checkb
ded_name[30]
ded_name1[30]
```

This file is used to download reconfigurable viewport information and trajectory tables, and it specifies the database and DED to be used.

2.1.2.2.1.8.1 use_print_checkb

This routine sets the value of *print_checkb* equal to 1, allowing *check_all()* to be used to check the buffer status after communications. It should be used only for debugging purposes.

2.1.2.2.1.8.2 set_ded_name

This routine changes the name of the DED from the default to *name*. The default name is *ded_otw.a301*.

Parameters		
Parameter	Type	Where Typedef Declared
name	pointer to char	Standard

Table 2.1-141: set_ded_name Information.

2.1.2.2.1.8.3 cig_reconfig_start

This routine notifies the CIG of the interface configurations and places the CIG in simulation mode. The interface configurations include: the database names, DED names, initial location, trajectory tables, viewport nodes, and configuration nodes.

Calls	
Function	Where Described
cig_receive_buffer	Section 2.1.2.2.1.9.1
flush_buffer	Section 2.1.2.2.2.28.1
push_msg_cig_ctl	Section 2.1.2.2.2.68.1
set_buffer_num	Section 2.1.2.2.2.113.1
cig_send_buffer	Section 2.1.2.2.1.10.1
check_all	Section 2.1.2.2.2.17.1
get_front_of_send_buffer	Section 2.1.2.2.2.32.1
multi_cig_prepend_dr11_pkt_size	Section 2.1.2.2.2.71.1
set_chunk_size	Section 2.1.2.2.3.23.1
push_msg_file_descr	Section 2.1.2.2.2.72.1
weapons_download_ballistics_table	Sections 2.2.3.2, and 2.3.3.2
cig_msg_configure_traj	Section 2.1.2.2.2.23.3
cig_setup_configuration	Section 2.1.2.2.1.11.1
cig_prepare_buffer	Sections 2.1.2.2.6.6, 2.1.2.2.7.5, and 2.1.2.2.8.4

Table 2.1-142: cig_reconfig_start Information.

2.1.2.2.1.9 cig_rcv_buf.c

(./simnet/release/src/vehicle/libsrc/libcig/cig_rcv_buf.c)

Includes:

"stdio.h"

"sim_dfns.h"

"rtc.h"
 "cig_local.h"
 "libcig.h"

Constant Defines:

C_RECV
 C_POLL
 C_ONE
 C_TWO

2.1.2.2.1.9.1 cig_receive_buffer

This routine is used to receive a buffer (interfacing with libcif), and is called every frame. The host is placed into either receive state or polling state. At the start of the frame, it is in receive state. First this routine checks for the number of CIGs, then it calls `cif_receive()` and waits for a return. If the return from `cif_receive()` is zero, the buffer was properly received. `cig_state` is then cleared to zero, and the routine is finished until the next frame. If the return from `cif_receive()` is non-zero, the host enters the polling state. During polling state, it continues working, but periodically checks for a return value from `cif_receive()`. `cig_state` may only be cleared to zero when the buffer has been received. If more than one CIG exists, the `cif_receive()` returns from both CIGs must be zero in order to clear `cig_state` to zero.

Internal Variables		
Variable	Type	Where Typedef Declared
cig_state	int	Standard
cig_mode	int	Standard
start_frame_time	int	Standard
ret1	int	Standard
ret2	int	Standard
temp	int	Standard
Return Values		
Return Value	Type	Meaning
cig_state	int	the current state of the CIG; - if zero, receive buffers are complete for all CIGs - if non-zero, receive buffers are not complete for one or more CIGs
Calls		
Function	Where Described	
bbd bit out	Section 2.1.5.1.4.1	
cif receive	Section 2.1.2.1.1.7.1	
get receive buffer	Section 2.1.2.2.1.21.1	
get receive size	Section 2.1.2.2.1.20.1	
rtc stop time	Section 2.6.16.1.3	
rtc start time	Section 2.6.16.1.2	

Table 2.1-143: cig_receive_buffer Information.

2.1.2.2.1.10 cig_send_buf.c**(./simnet/release/src/vehicle/libsrc/libcig/cig_send_buf.c)****Includes:**

"stdio.h"
"errno.h"
"sim_dfns.h"
"cig_local.h"
"libmsg.h"
"libcig.h"

Constant Defines:

C_SEND
C_POLL
C_ONE
C_TWO

Static Procedure Declarations:

cig_kickoff_dr_transfer()
cig_poll_dr_transfer()
cig_setup_dr_transfer()

Global Variable Declarations:

cig_state

2.1.2.2.1.10.1 cig_send_buffer

This routine is used to send a buffer (interfacing with libcif), and is called every frame. The host is set up in either send state or polling state. At the start of the frame, it is in send state. The transfer is initiated (or "kicked off") by calling **cig_kickoff_dr_transfer()**. The routine checks to ensure that the transfer was actually started, since occasionally the DR11 is not ready to send when the kickoff is called. If the transfer was initiated and the **cig_kickoff_dr_transfer()** call to **cif_send()** returns zero, the buffer was properly sent. **cig_state** is cleared to zero, and the routine is finished until the next frame. If the transfer was not initiated, the host loops and continues calling **cig_kickoff_dr_transfer()** and checking until the transfer is initiated. If kickoff occurred and the **cif_send()** return is non-zero, the host enters the polling state. During polling state, the host continues working, but periodically checks for a return value. **cig_state** may only be cleared when confirmation is received that the buffer has been sent.

Internal Variables		
Variable	Type	Where Typedef Declared
ret1	int	Standard
ret2	int	Standard
cig_mode	int	Standard
Return Values		
Return Value	Type	Meaning
cig_state	int	the current state of the CIG
Calls		
Function	Where Described	
cig_poll_dr_transfer	Section 2.1.2.2.1.10.3	
cig_setup_dr_transfer	Section 2.1.2.2.1.10.4	
cig_kickoff_dr_xfer	Section 2.1.2.2.1.10.2	
set_send_status	Section 2.1.2.2.1.25.2	
bbd_bit_out	Section 2.1.5.1.4.1	
get_send_size	Section 2.1.2.2.1.22.1	
cif_send	Section 2.1.2.1.1.8.1	
get_front_of_send_buffer	Section 2.1.2.2.2.32.1	

Table 2.1-144: **cig_send_buffer** Information.

2.1.2.2.1.10.2 cig_kickoff_dr_xfer

This routine checks for the number of CIGs and sends the buffer by calling `cif_send()` .

Parameters		
Parameter	Type	Where Typedef Declared
cig transfers	int	Standard
Internal Variables		
Variable	Type	Where Typedef Declared
errno	extern int	Standard
Calls		
Function	Where Described	
bbd bit out	Section 2.1.5.1.4.1	
get send size	Section 2.1.2.2.1.22.1	
cif send	Section 2.1.2.1.1.8.1	
get front of send buffer	Section 2.1.2.2.2.32.1	

Table 2.1-145: cig_kickoff_dr_xfer Information.

2.1.2.2.1.10.3 cig_poll_dr_transfer

This routine checks the return from `cif_send()` and clears the *cig_state* when the return is equal to zero.

Internal Variables		
Variable	Type	Where Typedef Declared
ret1	int	Standard
ret2	int	Standard
Calls		
Function	Where Described	
cif send	Section 2.1.2.1.1.8.1	
get front of send buffer	Section 2.1.2.2.2.32.1	
get send size	Section 2.1.2.2.1.22.1	

Table 2.1-146: cig_poll_dr_transfer Information.

2.1.2.2.1.10.4 cig_setup_dr_transfer

This routine places the overall header on the CIG message buffer being sent.

Calls	
Function	Where Described
cig msg append end	Section 2.1.2.2.2.4.1
cig msg prepend overall header	Section 2.1.2.2.2.86.1
set buffer num	Section 2.1.2.2.2.113.1

Table 2-147: cig_setup_dr_transfer Information.

2.1.2.2.1.11 cig_set_conf.c**(./simnet/release/src/vehicle/libsrc/libcig/cig_set_conf.c)****Includes:**

"stdio.h"
"sim_dfns.h"
"mass_stdc.h"
"dgi_stdg.h"
"sim_cig_if.h"
"libmsg.h"
"cig_local.h"

2.1.2.2.1.11.1 cig_setup_configuration

This routine instructs **cig_msg_configure_view()** in "libmsg" to put the CIG into CIG_CINFIG state, and then it reads in and processes the configuration file. It formats the CIG buffer accordingly.

Calls	
Function	Where Described
cig_msg_configure view	Section 2.1.2.2.23.4

Table 2.1-148: cig_setup_configuration Information.

2.1.2.2.1.12 cig_stop.c

(/simnet/release/src/vehicle/libsrc/libcig/cig_stop.c)

Includes:

```
"sim_types.h"
"stdio.h"
"mass_stdh.h"
"dgi_stdg.h"
"sim_cig_if.h"
"libmsg.h"
"libcig.h"
"cig_local.h"
```

2.1.2.2.1.12.1 cig_stop

This routine is used to place the CIG into the stop state, which means that it is idle. In order to stop the CIG, a check is first made for any buffer sends that are still pending. Once the sends are completed, the buffer is cleared. A message is then sent to the CIG informing it of the settings of the view flags, in order to ensure that the screens are turned off. All buffer pointers are reset, and a message is sent to the CIG to switch to the stop state. The DR11 transfer size is then reset to its initial size (512k in each direction).

Calls	
Function	Where Described
get_send_status	Section 2.1.2.2.1.25.1
cig_send_buffer	Section 2.1.2.2.1.10.1
cig_receive_buffer	Section 2.1.2.2.1.9.1
flush_buffer	Section 2.1.2.2.2.28.1
cig_msg_prepend_view_flags	Section 2.1.2.2.2.106.1
get_view_flags	Section 2.1.2.2.4.4.1
get_br_vals	Section 2.1.2.2.4.3.1
set_buffer_num	Section 2.1.2.2.2.113.1
buffer_reset	Section 2.1.2.2.2.15.1
push_msg_cig_ctl	Section 2.1.2.2.2.68.1
multi_cig_prepend_dr11_pkt_size	Section 2.1.2.2.2.79.1

Table 2.1-149: cig_stop Information.

2.1.2.2.1.13 cig_sync.c

(./simnet/release/src/vehicle/libsrc/libcig/cig_sync.c)

Includes:

"sim_types.h"
 "stdio.h"
 "errno.h"
 "mass_stdh.h"
 "dgi_stdg.h"
 "sim_cig_if.h"
 "libmsg.h"
 "libcig.h"
 "cig_local.h"

Variable Declarations:

size_of_header
 max_cig_transfer_size
 print_checkb

2.1.2.2.1.13.1 cig_synchronize

This routine synchronizes the CIG and the Simulation Host by first ensuring that all pending sends are completed, and then by performing ten read/write sequences. The parameter *ok_to_print* places the system into verbose mode, where status messages will be printed during operation.

Parameters		
Parameter	Type	Where Typedef Declared
ok_to_print	int	Standard
Internal Variables		
Variable	Type	Where Typedef Declared
i	int	Standard
Calls		
Function	Where Described	
get send status	Section 2.1.2.2.1.25.1	
cig send buffer	Section 2.1.2.2.1.10.1	
cig receive buffer	Section 2.1.2.2.1.9.1	
flush buffer	Section 2.1.2.2.2.28.1	
push msg cig ctl	Section 2.1.2.2.2.68.1	
set buffer num	Section 2.1.2.2.2.113.1	

Table 2.1-150: cig_synchronize Information.

2.1.2.2.1.14 cig_uninit.c

(/simnet/release/src/vehicle/libsrc/libcig/cig_uninit.c)

Includes:

```

"stdio.h"
"errno.h"
"sim_dfns.h"
"mass_stdh.h"
dgi_stdh.h"
"sim_cig_if.h"
"cig_local.h"

```

2.1.2.2.1.14.1 cig_uninit

This routine initializes the CIG by calling `cif_disconnect()` and `cif_uninit()`. The parameter `ok_to_print` checks if the system is in verbose mode.

Parameters		
Parameter	Type	Where Typedef Declared
ok_to_print	int	Standard
Errors		
Error	Reason for Error	
errno	<ul style="list-style-type: none">- cif_disconnect failed for cig1- cif_disconnect failed for cig2- cif_uninit failed	
Calls		
Function	Where Described	
cif_disconnect	Section 2.1.2.1.1.3.1	
cif_uninit	Section 2.1.2.1.1.9.1	

Table 2.1-151: cig_uninit Information.

2.1.2.2.1.15 cig_use_gra.c

(/simnet/release/src/vehicle/libsrc/libcig/cig_use_gra.c)

Includes:

```

"stdio.h"
"sim_dfns.h"
"mass_stdh.h"
dgi_stdh.h"
"sim_cig_if.h"
"cig_local.h"

```

2.1.2.2.1.15.1 cig_using_graphics

This routine sets the value of `using_graphics` equal to TRUE.

2.1.2.2.1.16 db_override.c

(./simnet/release/src/vehicle/libsrc/libcig/db_override.c)

Includes:

```
"stdio.h"
"sim_types.h"
"sim_dfns.h"
"mass_stdh.h"
"dgi_stdh.h"
"sim_cig_if.h"
"cig_local.h"
```

2.1.2.2.1.16.1 cig_use_database_override_named

This routine sets the database name to *db_name*, which will override the database specified by the activate packet.

Parameters		
Parameter	Type	Where Typedef Declared
db_name	pointer to char	Standard

Table 2.1-152: **cig_use_database_override_named** Information.

2.1.2.2.1.17 get_cig2.c

(./simnet/release/src/vehicle/libsrc/libcig/get_cig2.c)

Includes:

```
"stdio.h"
"sim_types.h"
"cig_local.h"
```

2.1.2.2.1.17 get_cig2_present

This routine checks if a second CIG is present.

Return Values		
Return Value	Type	Meaning
cig2_present	int	If TRUE, there is a second CIG present; If FALSE, only one CIG is present

Table 2.1-153: **get_cig2_present** Information.

2.1.2.2.1.18 get_i_sizes.c

(./simnet/release/src/vehicle/libsrc/libcig/get_i_sizes.c)

Includes:

"stdio.h"
 "sim_types.h"
 "sim_macros.h"
 "cig_local.h"

2.1.2.2.1.18.1 get_initial_sizes

This routine obtains the initial transfer size for the send and receive buffers via DR11 transfer. The default is 512 for the send buffer and 512 for the receive buffer.

Parameters		
Parameter	Type	Where Typedef Declared
init_send	pointer to int	Standard
init_rcv	pointer to int	Standard

Table 2.1-154: get_initial_sizes Information.

2.1.2.2.1.19 get_max.c

(./simnet/release/src/vehicle/libsrc/libcig/get_max.c)

Includes:

"stdio.h"
 "sim_types.h"
 "cig_local.h"

2.1.2.2.1.19.1 get_max_buffer_sizes

The maximum DR11 transfer sizes are given for the CIG specified in *bnum*. This routine is used by the CIG utility transfer programs. The maximum blocks of memory available from *cif_init()* for sending and receiving are pointed to by the parameters *send* and *rcv*.

Parameters		
Parameter	Type	Where Typedef Declared
bnum	int	Standard
send	pointer to int	Standard
rcv	pointer to int	Standard
Errors		
Error	Reason for Error	
stderr	Invalid buffer number	

Table 2.1-155: get_max_buffer_sizes Information.

2.1.2.2.1.20 get_r_size.c

(./simnet/release/src/vehicle/libsrc/libcig/get_r_size.c)

Includes:

"stdio.h"
 "sim_types.h"
 "cig_local.h"

2.1.2.2.1.20.1 get_receive_size

This routine obtains the receive buffer size.

Return Values		
Return Value	Type	Meaning
req_receive_size	int	the receive buffer size if the default value was not used
initial_receive_size	int	the receive buffer size if the default value was used

Table 2.1-156: get_receive_size Information.

2.1.2.2.1.21 get_rcv_buf.c

(./simnet/release/src/vehicle/libsrc/libcig/get_rcv_buf.c)

Includes:

"stdio.h"
 "sim_types.h"
 "cig_local.h"

2.1.2.2.1.21.1 get_receive_buffer

This routine returns a pointer to the beginning of the receive buffer for the CIG specified in *cig_num*.

Parameters		
Parameter	Type	Where Typedef Declared
cig_num	int	Standard
Return Values		
Return Value	Type	Meaning
&receive_buf[cig_num-1]	pointer to pointer to char	pointer to the beginning of the receive buffer

Table 2.1-157: get_receive_buffer Information.

2.1.2.2.1.22 get_s_size.c

(./simnet/release/src/vehicle/libsrc/libcig/get_s_size.c)

Includes:

"stdio.h"
 "sim_types.h"
 "cig_local.h"

2.1.2.2.1.22.1 get_send_size

This routine obtains the send buffer size.

Return Values		
Return Value	Type	Meaning
req_send_size	int	the send buffer size if the initial value was not used
initial_send_size	int	the send buffer size if the initial value was used

Table 2.1-158: get_send_size Information.

2.1.2.2.1.23 not_prep_buf.c

(./simnet/release/src/vehicle/libsrc/libcig/not_prep_buf.c)

Includes:

"stdio.h"
 "sim_types.h"
 "mass_std.h"
 "dgi_stdg.h"
 "sim_cig_if.h"
 "cig_local.h"

2.1.2.2.1.23.1 cig_not_ok_to_prepare_buffer

This routine determines if the buffer should be prepared (i.e., if the CIG is being used) by checking the *using_graphics* flag. This routine is used for debugging purposes.

Return Values		
Return Value	Type	Meaning
FALSE	int	CIG is being used
TRUE	int	CIG is not being used

Table 2.1-159: cig_not_ok_to_prepare_buffer Information.

2.1.2.2.1.24 not_proc_buf.c

(/simnet/release/src/vehicle/libsrc/libcig/not_proc_buf.c)

Includes:

```

"stdio.h"
"sim_types.h"
"mass_stdh.h"
"dgi_stdg.h"
"sim_cig_if.h"
"cig_local.h"

```

2.1.2.2.1.24.1 cig_not_ok_to_process_buffer

This routine determines if the buffer should be processed (i.e., if the CIG is being used) by checking the *using_graphics* flag. This routine is used for debugging purposes.

Return Values		
Return Value	Type	Meaning
FALSE	int	CIG is being used
TRUE	int	CIG is not being used

Table 2.1-160: cig_not_ok_to_process_buffer Information.

2.1.2.2.1.25 send_status.c

(/simnet/release/src/vehicle/libsrc/libcig/send_status.c)

Includes "libcig.h"

Buffer Declaration:

send_status

This file contains routines for communicating cig_send completion status between the invoker of a send to the CIG(s) and net_simul (who must determine if the send has completed before beginning to build a new CIG buffer).

2.1.2.2.1.25.1 get_send_status

This routine determines the status of the send buffer (i.e., complete or pending).

Return Values		
Return Value	Type	Meaning
send_status	int	the status of the send buffer (either complete or pending)

Table 2.1-161: get_send_status Information.

2.1.2.2.1.25.2 set_send_status

This routine sets the status of the send buffer to *s*.

Parameters		
Parameter	Type	Where Typedef Declared
<i>s</i>	int	Standard
Return Values		
Return Value	Type	Meaning
send_status = <i>s</i>	int	the status of the send buffer (either complete or pending)

Table 2.1-162: set_send_status Information.

2.1.2.2.1.26 set_cig_dev.c

(./simnet/release/src/vehicle/libsrc/libcig/set_cig_dev.c)

Includes:

"stdio.h"
 "sim_types.h"
 "sim_dfns.h"
 "cig_local.h"

2.1.2.2.1.26.1 set_cig_dev

This routine sets the device number for each CIG. *cig_num* is the CIG being set (either CIG1 or CIG2) and *dev_num* is the device number being assigned to it.

Parameters		
Parameter	Type	Where Typedef Declared
<i>cig_num</i>	int	Standard
<i>dev_num</i>	int	Standard

Table 2.1-163: set_cig_dev Information.

2.1.2.2.1.27 set_i_sizes.c

(/simnet/release/src/vehicle/libsrc/libcig/set_i_sizes.c)

Includes:

```
"stdio.h"
"sim_types.h"
"sim_macros.h"
"cig_local.h"
```

2.1.2.2.1.27.1 set_initial_sizes

This routine allows the initial transfer size to be specified for the send and receive buffers via DR11 transfer. *init_send* is the initial transfer size of the send buffer, and *init_rcv* is the initial transfer size of the receive buffer. The default sizes are both 512.

Parameters		
Parameter	Type	Where Typedef Declared
init_send	int	Standard
init_rcv	int	Standard
Errors		
Error	Reason for Error	
stderr	- Exceeded max possible send initializations - Exceeded max possible rcv initializations	

Table 2.1-164: set_initial_sizes Information.

2.1.2.2.1.28 set_my_if.c

(/simnet/release/src/vehicle/libsrc/libcig/set_my_if.c)

Includes:

```
"stdio.h"
"sim_types.h"
"cig_local.h"
```

2.1.2.2.1.28.1 set_my_if

This routine sets the interface number to *i_num*.

Parameters		
Parameter	Type	Where Typedef Declared
i_num	int	Standard

Table 2.1-165: set_my_if Information.

2.1.2.2.1.29 set_req_rcv.c

(./simnet/release/src/vehicle/libsrc/libcig/set_req_rcv.c)

Includes:

"stdio.h"
"sim_types.h"
"cig_local.h"

2.1.2.2.1.29.1 set_request_receive_size

This routine sets the receive buffer size to the value in *req_size*.

Parameters		
Parameter	Type	Where Typedef Declared
req_size	int	Standard

Table 2.1-166: set_request_receive_size Information.

2.1.2.2.1.30 set_req_send.c

(./simnet/release/src/vehicle/libsrc/libcig/set_req_send.c)

Includes:

"stdio.h"
"sim_types.h"
"cig_local.h"

2.1.2.2.1.30.1 set_request_send_size

This routine sets the send buffer size to the value in *req_size*.

Parameters		
Parameter	Type	Where Typedef Declared
req_size	int	Standard

Table 2.1-167: set_request_send_size Information.

2.1.2.2.1.31 set_s_flag.c**(./simnet/release/src/vehicle/libsrc/libcig/set_s_flag.c)****Includes:**

"stdio.h"
"sim_types.h"
"cig_local.h"

2.1.2.2.1.31.1 set_use_requested_flag

This routine allows the boolean status of *use_requested_sizes* to be changed. If *bool* = 1, the transfer size will be the size requested in either *set_request_send_size()* or *set_request_receive_size()*; otherwise, the transfer size equals the initial size.

Parameters		
Parameter	Type	Where Typedef Declared
bool	int	Standard

Table 2.1-168: set_use_requested_flag Information.

2.1.2.2.1.32 setup_buf.c**(./simnet/release/src/vehicle/libsrc/libcig/setup_buf.c)****Includes:**

"stdio.h"
"sim_dfns.h"
"mass_stdh.h"
"dgi_stdg.h"
"sim_cig_if.h"
"libmsg.h"
"gbuffer.h"
"cig_buffer.h"
"cig_local.h"

Defines:

OTHERVEH_OFFSET

2.1.2.2.1.32.1 setup_buffer_ptrs

This routine sets up the following fixed buffer pointers in *s_buf*: *start_of_send_buffer*, *end_of_send_buffer*, and *other_start_of_send_buffer*. *num* specifies the buffer, and *buf* points to the beginning of the buffer.

Parameters		
Parameter	Type	Where Typedef Declared
num	int	Standard
buf	pointer to char	Standard
Internal Variables		
Variable	Type	Where Typedef Declared
s_buf	register pointer to S_BUFFER	Section 2.1.2.2.117
Calls		
Function	Where Described	
get_init_ptrs	Section 2.1.2.2.33.1	

Table 2.1-169: setup_buffer_ptrs Information.

2.1.2.2.1.33 cig_get_db.c

(./simnet/release/src/vehicle/libsrc/libcig/cig_get_db.c)

Includes "cig_local.h"

2.1.2.2.1.33.1 cig_get_db

This routine returns a pointer to the start of the database name.

Return Values		
Return Value	Type	Meaning
database_name	pointer to char	the database name

Table 2.1-170: cig_get_db Information.

2.1.2.2.2 libmsg (./simnet/release/src/vehicle/libsrc/libmsg [libmsg])

Libmsg contains the routines that are called when information is being formatted in the buffer for transfer to the CIG.

2.1.2.2.2.1 add_veh2cig.c (./simnet/release/src/vehicle/libsrc/libmsg/add_veh2cig.c)

The routine in this file adds a dynamic vehicle to the CIG buffer.

Includes:

```
"stdio.h"  
"sim_dfns.h"  
"sim_macros.h"  
"sim_types.h"  
"sines.h"  
"mass_stdh.h"  
"dgi_stdg.h"  
"sim_cig_if.h"  
"libveh.h"  
"librva.h"  
"libmsg.h"  
"libmap.h"
```

2.1.2.2.2.1.1 add_veh_to_cig_msg

This routine adds a dynamic vehicle to the CIG buffer, where *r* is a pointer to an entry in the RVA table. It returns TRUE if the vehicle has been added and returns FALSE otherwise.

Parameters		
Parameter	Type	Where Typedef Declared
<i>r</i>	pointer to RVA_ENTRY	librva.h
Internal Variables		
Internal Variable	Type	Where Typedef Declared
<i>i</i>	register int	Standard
<i>mp</i>	pointer to MAG_OTHERVEH_STATE	sim_cig_if.h
<i>from</i>	register pointer to float	Standard
<i>to</i>	register pointer to REAL_4	mass_std.h
<i>sine_ptr</i>	register pointer to float	Standard
<i>sin_cos_index</i>	int	Standard
Return Values		
Return Value	Type	Meaning
TRUE	int	vehicle added
FALSE	int	can't add vehicle
Calls		
Function	Where Described	
append_other_in_send_buffer	Section 2.1.2.2.2.13.1	
network_get_vehicle_force	Section 2.1.1.3.1.17.1	
map_net_to_cig	Section 2.6.11.5.7	
SINES_SHIFT_INDEX	Appendix A sines.h (macro definition)	

Table 2.1-171: add_veh_to_cig_msg Information.

2.1.2.2.2.2 adj_chg_stat.c

(/simnet/release/src/vehicle/libsrc/libmsg/adj_chg_stat.c)

This file contains routines which are used to modify the static vehicle information in the CIG buffer.

Includes:

```
"stdio.h"
"sim_types.h"
"mass_stdh.h"
"dgi_stdg.h"
"sim_cig_if.h"
"libveh.h"
"librva.h"
"libmsg.h"
"libmap.h"
"pro_sim.h"
"msg_loc.h"
```

2.1.2.2.2.2.1 fill_changed_static_remove_msg

This routine fills the Static Vehicle Remove message, which informs the CIG of which static vehicles to remove from the display. *mp* is a pointer to the Static Vehicle Remove message, and *pkt* is a pointer to the Vehicle Appearance Variant.

Parameters		
Parameter	Type	Where Typedef Declared
mp	pointer to MSG_STATICVEH_REM	sim_cig_if.h
pkt	pointer to VehicleAppearanceVariant	p_sim.h
Internal Variables		
Internal Variable	Type	Where Typedef Declared
from	register pointer to REAL	sim_types.h
to	register pointer to REAL 4	mass_stdh.h
r_pkt	pointer to RVA_ENTRY	librva.h
Calls		
Function	Where Described	
network_get_vehicle_force	Section 2.1.1.3.1.17.1	
map_net_to_cig	Section 2.6.11.5.7	

Table 2.1-172: fill_changed_static_remove_msg Information.

2.1.2.2.2.2.2 fill_changed_static_msg

This routine fills the Static Vehicle State message. *mp* is a pointer to the Static Vehicle State message, and *pkt* is a pointer to the Vehicle Appearance Variant.

Parameters		
Parameter	Type	Where Typedef Declared
mp	pointer to MSG_STATICVEH_STATE	sim_cig_if.h
pkt	pointer to VehicleAppearanceVariant	p_sim.h
Internal Variables		
Internal Variable	Type	Where Typedef Declared
from	register pointer to REAL	sim_types.h
to	register pointer to REAL_4	mass_std.h
r_pkt	pointer to RVA_ENTRY	librva.h
Calls		
Function	Where Described	
network_get_vehicle_force	Section 2.1.1.3.1.17.1	
map_format_asid	Section 2.6.11.4.6	
map_net_to_cig	Section 2.6.11.5.7	

Table 2.1-173: fill_changed_static_msg Information.

2.1.2.2.2.3 cig_adjust_for_changed_staticveh

Information about static vehicles is sent to the CIG when they first enter the viewing range. Static vehicles are destroyed or restored to health periodically throughout the simulation. This routine allocates buffer space and fills in the appropriate information for the static vehicles which have changed in appearance.

Parameters		
Parameter	Type	Where Typedef Declared
veh_list[]	pointer to RVA_ENTRY	librva.h
num_vehs	int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
pkt	pointer to VehicleAppearanceVariant	p_sim.h
mp_rem	register pointer to MSG_STATICVEH_REM	sim_cig_if.h
mp_add	register pointer to MSG_STATICVEH_STATE	sim_cig_if.h
k	register int	Standard
r_pkt	pointer to RVA_ENTRY	librva.h
Calls		
Function	Where Described	
prepend_msg_hdr	Section 2.1.2.2.2.83.1	
fill_changed_static_remove_msg	Section 2.1.2.2.2.1	
fill_changed_static_msg	Section 2.1.2.2.2.2	
deallocate_prependded_buffer_space	Section 2.1.2.2.2.26.1	

Table 2.1-174: cig_adjust_for_changed_staticveh Information.

2.1.2.2.2.3 adj_othervch.c

(/simnet/release/src/vehicle/libsrc/libmsg/adj_othervch.c)

Includes:

```

"stdio.h"
"mass_stdh.h"
"sim_types.h"
"dgi_stdh.h"
"sim_cig_if.h"
"libveh.h"
"librva.h"
"gbuffer.h"
"pro_sim.h"

```

2.1.2.2.2.3.1 cig_msg_adjust_othervch_state

This routine adjusts the Other Vehicle State message. Information about the location of dynamic vehicles is sent to the CIG every frame. *num_veh*s is the number of dynamic vehicles which exist in the CIG buffer, and *veh_list* is a pointer to the RVA list which contains vehicle location information.

Parameters		
Parameter	Type	Where Typedef Declared
veh_list[]	pointer to RVA_ENTRY	librva.h
num_veh	int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
pkt	pointer to VehicleAppearanceVariant	p_sim.h
from	register pointer to REAL	sim_types.h
k	register int	Standard
to	register pointer to REAL_4	mass_stdh.h

Table 2.1-175: cig_msg_adjust_othervch_state Information.

2.1.2.2.2.4 app_end.c

(./simnet/release/src/vehicle/libsrc/libmsg/app_end.c)

Includes:

```
"stdio.h"
"sim_macros.h"
"sim_dfns.h"
"mass_stdh.h"
"sim_types.h"
"dgi_stdg.h"
"sim_cig_if.h"
"gbuffer.h"
"msg_loc.h"
```

2.1.2.2.2.4.1 cig_msg_append_end

This routine appends an end message to the CIG buffer. This message has no body. It consists of a message header only. Room should be left in the message buffer for the end message. This routine obtains its own space from the buffer because the routine **append_msg_hdr()** always reserves room for an M_END, even if the message which is being appended is an end message.

Internal Variables		
Internal Variable	Type	Where Typedef Declared
hp	register pointer to MSG HDR	sim_cig_if.h
Calls		
Function	Where Described	
get_send_size	Section 2.1.2.2.1.22.1	
REPORT_ERROR	gbuffer.h (macro definition)	

Table 2.1-176: cig_msg_append_end Information.

2.1.2.2.2.5 app_msg_hdr.c

(./simnet/release/src/vehicle/libsrc/libmsg/app_msg_hdr.c)

The routine in this file appends the message header.

Includes:

```
"stdio.h"
"sim_types.h"
"mass_stdh.h"
"dgi_stdh.h"
"sim_cig_if.h"
"gbuffer.h"
"msg_loc.h"
```

2.1.2.2.2.5.1 append_msg_hdr

This routine is called to request the next piece of memory on the bottom of the current buffer. If there is enough memory available, it allocates *length* bytes of space, plus a message header which will be formatted to specify the type of the message. If this routine is unsuccessful, it returns NULL; otherwise, it returns a pointer to the memory that the caller should use for the message.

This routine checks two different pointers to determine if there is still enough of the buffer available. It first checks the bottom of the total buffer allocated (*end_of_send_buffer*) to ensure that there is enough room for the MSG_HDR that will be used for the MSG_END. It also checks against the current top of the send buffer (*front_of_send_buffer*) to ensure that there will be space for both the MSGS_BLK header and the MSG_HDR that will be used for the MSG_END.

Parameters		
Parameter	Type	Where Typedef Declared
type	int	Standard
length	int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
hp	register pointer to MSG_HDR	sim_cig_if.h
Return Values		
Return Value	Type	Meaning
NULL	pointer to char	invalid message type; not enough space in the message buffer
s_buffer_ptrs[buf_num].back of_send_buffer-length	pointer to char	pointer to memory that should be used for the message
Calls		
Function	Where Described	
get send size	Section 2.1.2.2.1.22.1	
ERROR REPORT	gbuffer.h (macro definition)	

Table 2.1-177: append_message_hdr Information.

2.1.2.2.2.6 app_mtra_ent.c (./simnet/release/src/vehicle/libsrc/libmsg/app_mtra_ent.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_stdh.h"
"dgi_stdh.h"
"sim_cig_if.h"
"msg_loc.h"
"libmsg.h"
```

2.1.2.2.2.6.1 multi_cig_append_traj_entry_xfer

This routine appends the Trajectory Entry Transfer message to the back of the message buffer. This routine is used for a system with two CIGs.

The parameters are defined as follows:

buf_mask - indicates CIG buffer.
bore_x - the X-coordinate of the trajectory with respect to the gun barrel.
bore_z - the Z-coordinate of the trajectory with respect to the gun barrel.

Parameters		
Parameter	Type	Where Typedef Declared
<i>buf_mask</i>	int	Standard
<i>bore_x</i>	REAL	sim_types.h
<i>bore_z</i>	REAL	sim_types.h
Calls		
Function	Where Described	
<i>cig_msg_append_traj_entry_xfer</i>	Section 2.1.2.2.2.10.1	

Table 2.1-178: multi_cig_append_traj_entry_xfer Information.

2.1.2.2.2.7 app_mtra_tbl.c (./simnet/release/src/vehicle/libsrc/libmsg/app_mtra_tbl.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_std.h"
"dgi_stdg.h"
"sim_cig_if.h"
"msg_loc.h"
"libmsg.h"
```

2.1.2.2.2.7.1 multi_cig_append_traj_table_xfer

This routine appends the Trajectory Table Transfer message to the back of the message buffer. This routine is only called for a system with two CIGs.

The parameters are defined as follows:

<i>buf_mask</i>	- indicates CIG buffer.
<i>ammo_type</i>	- the ammunition type to be modeled.
<i>traj_index</i>	- the trajectory table number, which specifies a given trajectory table.
<i>count</i>	- the size of the trajectory table.

Parameters		
Parameter	Type	Where Typedef Declared
<i>buf_mask</i>	int	Standard
<i>ammo_type</i>	int	Standard
<i>traj_index</i>	int	Standard
<i>count</i>	int	Standard
Calls		
Function	Where Described	
<i>cig_msg_append_traj_table_xfer</i>	Section 2.1.2.2.2.11.1	

Table 2.1-179: multi_cig_append_traj_table_xfer Information.

2.1.2.2.2.8 app_stat_rm.c

(/simnet/release/src/vehicle/libsrc/libmsg/app_stat_rm.c)

Includes:

```

"stdio.h"
"sim_dfns.h"
"sim_macros.h"
"sim_types.h"
"mass_stdh.h"
"dgi_stdg.h"
"sim_cig_if.h"
"libveh.h"
"pro_sim.h"
"librva.h"
"libmsg.h"
"msg_loc.h"
"libmap.h"

```

2.1.2.2.2.8.1 cig_msg_append_staticveh_rem

This routine appends the Static Vehicle Remove message to the end of the message buffer. *veh_list* is a pointer to the Vehicle Appearance Variant, which contains information on the appearance and location of the vehicle. The number of static vehicles to be removed is indicated by *num_vchs*.

Parameters		
Parameter	Type	Where Typedef Declared
veh_list	pointer to VehicleAppearanceVariant	p_sim.h
num vchs	int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
pkt	pointer to VehicleAppearanceVariant	p_sim.h
mp	register pointer to MSG_STATICVEH_REM	sim_cig_if.h
k	register int	Standard
from	register pointer to REAL	sim_types.h
to	register pointer to REAL 4	mass_stdh.h
Calls		
Function	Where Described	
network get vehicle force	Section 2.1.1.3.1.17.1	
map net to cig	Section 2.6.11.5.7	
append msg_hdr	Section 2.1.2.2.2.5.1	

Table 2.1-180: **cig_msg_append_staticveh_rem** Information.

2.1.2.2.2.9 app_stat_veh.c

(/simnet/release/src/vehicle/libsrc/libmsg/app_stat_veh.c)

Includes:

```

"stdio.h"
"sim_dfns.h"
"sim_macros.h"
"sim_types.h"
"mass_stdc.h"
"dgi_stdg.h"
"sim_cig_if.h"
"libveh.h"
"pro_sim.h"
"sibrva.h"
"libmsg.h"
"msg_loc.h"
"libmap.h"

```

2.1.2.2.2.9.1 cig_msg_append_staticveh_state

This routine appends the Static Vehicle State message to the end of the message buffer. *veh_list* is a pointer to the Vehicle Appearance Variant, which contains information on the appearance and location of the vehicles. The number of static vehicles in the buffer is indicated by *num_veh*s.

Parameters		
Parameter	Type	Where Typedef Declared
veh_list	pointer to VehicleAppearanceVariant	p_sim.h
num vehs	int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
pkt	pointer to VehicleAppearanceVariant	p_sim.h
mp	register pointer to MSG_STATICVEH_REM	sim_cig_if.h
k	register int	Standard
from	register pointer to REAL	sim_types.h
to	register pointer to REAL 4	mass_stdc.h
Calls		
Function	Where Described	
network_get_vehicle_force	Section 2.1.1.3.1.17.1	
map_net_to_cig	Section 2.6.11.5.7	
map_format_asid	Section 2.6.11.4.6	
append_message_hdr	Section 2.1.2.2.2.5.1	

Table 2.1-181: cig_msg_append_staticveh_state Information.

2.1.2.2.2.10 app_traj_ent.c

(./simnet/release/src/vehicle/libsrc/libmsg/app_traj_ent.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_std.h"
"dgi_std.h"
"sim_cig_if.h"
"libmsg.h"
```

2.1.2.2.2.10.1 cig_msg_append_traj_entry_xfer

This routine appends the Trajectory Entry Transfer message to the end of the CIG buffer. The X-coordinate of the trajectory with respect to the gun barrel is represented by *bore_x*. The Z-coordinate of the trajectory with respect to the gun barrel is represented by *bore_z*.

Parameters		
Parameter	Type	Where Typedef Declared
bore x	REAL	sim_types.h
bore z	REAL	sim_types.h
Internal Variables		
Internal Variable	Type	Where Typedef Declared
mpto	register pointer to MSG TRAJ ENTRY XFER	sim_cig_if.h
Calls		
Function	Where Described	
append message_hdr	Section 2.1.2.2.2.5.1	

Table 2.1-182: cig_msg_append_traj_entry_xfer Information.

2.1.2.2.2.11 app_traj_tbl.c

(./simnet/release/src/vehicle/libsrc/libmsg/app_traj_tbl.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_std.h"
"dgi_std.h"
"sim_cig_if.h"
"libmsg.h"
```

2.1.2.2.2.11.1 cig_msg_append_traj_table_xfer

This routine appends the Trajectory Table Transfer message to the end of the message buffer.

The parameters represent information that is to be added to the message buffer, and are defined as follows:

ammo_type - the ammunition type to be modeled.
traj_index - the trajectory table number.
count - the size of the trajectory table.

Parameters		
Parameter	Type	Where Typedef Declared
ammo_type	int	Standard
traj_index	int	Standard
count	int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
mp	register pointer to MSG TRAJ TABLE XFER	sim_cig_if.h
Calls		
Function	Where Described	
append_message_hdr	Section 2.1.2.2.5.1	

Table 2.1-183: cig_msg_append_traj_table_xfer Information.

2.1.2.2.2.12 app_vflags.c

(./simnet/release/src/vehicle/libsrc/libmsg/app_vflags.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_std.h"
"dgi_std.h"
"sim_cig_if.h"
"libvflags.h"
"libmsg.h"
"libmsg.h"
"msg_loc.h"
```

2.1.2.2.2.12.1 cig_msg_append_view_flags

This routine appends the View Flags message to the back of the CIG buffer. The parameters represent information that is to be added to the message buffer, and are defined as follows:

- view_flags* - values that turn on or off individual processing paths in the CIG system.
- branch_values* - an array of values that control the branching of branch nodes. Configuration branch nodes use values from this array to determine which branch to take. These values are compared against the branch mask specified in the Create Configuration Node message. Only the lower 24 bits of each branch value are used in the expression.

Parameters		
Parameter	Type	Where Typedef Declared
view flags	WORD	mass_std.h
branch value	WORD	mass_std.h
Internal Variables		
Internal Variable	Type	Where Typedef Declared
mp	register pointer to MSG_VIEW_FLAGS	sim_cig.h
Calls		
Function	Where Described	
append_msg_hdr	Section 2.1.2.2.5.1	

Table 2.1-184: cig_msg_append_view_flags Information.

2.1.2.2.2.13 append_other.c

(./simnet/release/src/vehicle/libsrc/libmsg/append_other.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_std.h"
"dgi_std.h"
"sim_cig_if.h"
"gbuffer.h"
"msg_loc.h"
```

2.1.2.2.2.13.1 append_other_in_send_buffer

This routine appends an Other Vehicle State message to the message buffer. Space is allocated and the information is provided for a new dynamic vehicle. The pointer *hp* points to the message header.

Internal Variables		
Internal Variable	Type	Where Typedef Declared
hp	register pointer to MSG_HDR	sim_cig_if.h
Return Values		
Return Value	Type	Meaning
NULL	pointer to char	back not equal to other_back out of space
s_buffer_ptrs[0].other_back_in_send_buffer-sizeof(MSG_OTHERVEH_STATE)	pointer to char	the location of the Other Vehicle State message in the buffer
Calls		
Function	Where Described	
ERROR_REPORT	gbuffer.h (macro definition)	

Table 2.1-185: append_other_in_send_buffer Information.

2.1.2.2.2.14 ball_buffer.c

(./simnet/release/src/vehicle/libsrc/libmsg/ball_buffer.c)

This file contains routines which create temporary buffer space. Sending a CIG buffer takes longer than a single frame; therefore, the actual simulation frame is overlapped with sending the CIG buffer from the previous frame. It is necessary to create some temporary buffer space for the trajectory chord messages and the round fired messages that have been generated before the previous buffer has been fully transferred. The maximum number of ballistics messages that can be sent are four trajectory chord messages and one round fired message.

This file was originally intended for ballistics messages. It is now used for any information that is queued up for the CIG outside of `cig_prepare_buffer()`.

Includes:

```
"stdio.h"
"sim_types.h"
"sim_dfns.h"
"dgi_stdh.h"
"dgi_stdg.h"
"sim_cig_if.h"
"if_ctas.h"
"gbuffer.h"
"msg_loc.h"
```

The following is defined:

BALL_BUFFER_SIZE

The following is declared:

ballistics_buffer[BALL_BUFFER_SIZE]

2.1.2.2.2.14.1 init_ballistics_buffer

This routine initializes the ballistics buffer.

2.1.2.2.2.14.2 copy_ballistics_buffer

This routine copies the temporary ballistics buffer to the CIG buffer. It assumes that, since there can be more than one CIG buffer, all buffers would want to be aware of the ballistics messages; therefore, it copies the ballistics information into the buffer associated with CIG 1. It also assumes that there is enough space in the CIG buffer to accomplish this task, because it is done very early in **cig_prepare_buffer()**. Nothing has been prepended at this point, so there will always be enough room for the ballistics messages.

Internal Variables		
Internal Variable	Type	Where Typedef Declared
copy_size	int	Standard

Table 2.1-186: copy_ballistics_buffer Information.

2.1.2.2.2.14.3 store_traj_chord

This routine adds the trajectory chord data to the ballistics buffer. The parameters represent information that will be placed in the message buffer, and are defined as follows:

- type* - the chord type, generally representing an ammunition or missile type.
- id* - an identifier for the chord.
- tracer* - a value of zero will prevent a tracer from being displayed; otherwise, the value represents the effect type, to be displayed as a tracer.
- begin* - the X-, Y-, and Z-coordinates describing the starting position of the chord.
- end* - the X-, Y-, and Z-coordinates describing the ending position of the chord.

Parameters		
Parameter	Type	Where Typedef Declared
type	int	Standard
id	int	Standard
tracer	BOOLEAN	sim_types.h
begin	VECTOR	sim_types.h
end	VECTOR	sim_types.h
Internal Variables		
Internal Variable	Type	Where Typedef Declared
hp	register pointer to MSG HDR	sim_cig_if.h
mp	register pointer to MSG TRAJ CHORD	sim_cig_if.h
time count	static int	Standard
Calls		
Function	Where Described	
ERROR_REPORT	gbuffer.h (macro definition)	

Table 2.1-187: store_traj_chord Information.

2.1.2.2.2.14.4 store_round_fired

This routine stores process round data in the ballistics buffer. The parameters represent information that will be stored in the buffer, and are defined as follows:

- type* - round type; specifies the trajectory table to use.
- tracer* - A value of 0 will cause the tracer effect type in the trajectory table specified by round type to be displayed as the tracer; otherwise, a different, valid tracer effect must be specified.
- id* - unique round identifier.
- gunpos* - the position of the gun tip, in world coordinates, at the time of firing.
- gunvel* - the elevation of the gun, in world coordinates, at the time of firing.
- sinelv* - the sine of the gun elevation angle, in world coordinates, at the time of firing.
- coselv* - the cosine of the gun elevation angle, in world coordinates, at the time of firing.
- sinazm* - the sine of the gun azimuth angle, in world coordinates, at the time of firing.
- cosazm* - the cosine of the gun azimuth angle, in world coordinates, at the time of firing.
- est_impact_time* - estimated impact time.
- est_impact_range* - estimated impact range.

Parameters		
Parameter	Type	Where Typedef Declared
<i>type</i>	int	Standard
<i>tracer</i>	int	Standard
<i>id</i>	int	Standard
<i>gunpos</i>	VECTOR	sim_types.h
<i>gunvel</i>	VECTOR	sim_types.h
<i>sinelv</i>	REAL	sim_types.h
<i>coselv</i>	REAL	sim_types.h
<i>sinazm</i>	REAL	sim_types.h
<i>cosazm</i>	REAL	sim_types.h
<i>est_impact_time</i>	REAL	sim_types.h
<i>est_impact_range</i>	REAL	sim_types.h
Internal Variables		
Internal Variable	Type	Where Typedef Declared
<i>hp</i>	register pointer to MSG_HDR	sim_cig_if.h
<i>mp</i>	register pointer to MSG_PROCESS_ROUND	sim_cig_if.h
Calls		
Function	Where Described	
REPORT_ERROR	gbuffer.h (macro definition)	

Table 2.1-188: store_round_fired Information.

2.1.2.2.2.14.5 store_view_magnification

This routine stores magnified view data in the temporary buffer. The parameters represent information that will be stored in the buffer, and are defined as follows:

- node_index* - the configuration node id to which the viewport is attached.
- lod_multiplier* - the level-of-detail multiplier; it adjusts the level of detail range.
- i* - specifies the horizontal field of view, in degrees.
- j* - specifies the vertical field of view, in degrees.

Parameters		
Parameter	Type	Where Typedef Declared
node_index	HWND	mass_std.h
lod_multiplier	REAL 4	mass_std.h
i	REAL 4	mass_std.h
j	REAL 4	mass_std.h
Internal Variables		
Internal Variable	Type	Where Typedef Declared
hp	register pointer to MSG_HDR	sim_cig_if.h
mp	register pointer to MSG_VIEW_MAGNIFICATION	sim_cig_if.h
Calls		
Function	Where Described	
REPORT_ERROR	gbuffer.h (macro definition)	

Table 2.1-189: store_view_magnification Information.

2.1.2.2.2.14.6 store_other_veh_state

This routine creates an Other Vehicle State message in the temporary message buffer. It is a debugging tool for the ctas head /engine tracker and is not used in this version of the code.

2.1.2.2.2.14.7 store_ctas_init_startup_model

This routine is not implemented in this version of the software.

2.1.2.2.2.14.8 store_ctas_grow_model

This routine is not implemented in this version of the software.

2.1.2.2.2.15 buf_reset.c

(./simnet/release/src/vehicle/libsrc/libmsg/buf_reset.c)

This file contains a routine which resets the CIG buffer.

Includes:

"sim_types.h"
"mass_stdh.h"
"dgi_stdh.h"
"sim_cig_if.h"
"gbuffer.h"
"cig_buffer.h"
"libmsg.h"
"msg_loc.h"

2.1.2.2.2.15.1 buffer_reset

This routine resets all of the pointers to the CIG buffer.

Calls	
Function	Where Described
clear n mapped	Section 2.1.2.2.20.1

Table 2.1-190: buffer_reset Information.

2.1.2.2.2.16 buf_setup.c

(./simnet/release/src/vehicle/libsrc/libmsg/buf_setup.c)

This file contains a routine which sets up the CIG buffer.

Includes:

"sim_types.h"
"mass_stdh.h"
"dgi_stdh.h"
"sim_cig_if.h"
"gbuffer.h"
"cig_buffer.h"
"libmsg.h"
"msg_loc.h"

2.1.2.2.2.16.1 buffer_setup

This routine resets all of the pointers to the CIG buffer. It is similar to **buffer_reset()**, but it allows the use of the entire 4k buffer.

2.1.2.2.2.17 check_all.c

(./simnet/release/src/vehicle/libsrc/libmsg/check_all.c)

Includes:

```
"sim_types.h"
"dgi_stdh.h"
"dgi_stdg.h"
"sim_cig_if.h"
"msg_loc.h"
"gbuffer.h"
```

The following external declaration is made:

```
print_checkb
```

2.1.2.2.2.17.1 check_all

This routine is called to check the message buffer. As the buffer is checked to ensure valid message types and correct information, the messages are printed out. This routine is used as a debugging tool. *mbp* is a pointer to the message buffer. *bnum* is the buffer identifier.

Parameters		
Parameter	Type	Where Typedef Declared
mbp	pointer to MSGS_BLK	sim_cig_if.h
bnum	int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
bytes_done	int	Standard
found_end	int	Standard
hp	MSG_HDR	sim_cig_if.h
error_found	int	Standard
Return Values		
Return Value	Type	Meaning
error_found	BOOLEAN	TRUE - an error in the buffer was found FALSE - no error was found
Calls		
Function	Where Described	
DOT	gbuffer.h (macro definition)	
print_msg_cig_ctl	Section 2.1.2.2.2.41.1	

Table 2.1-191: check_all Information.

2.1.2.2.2.18 checkbuffer.c

(./simnet/release/src/vehicle/libsrc/libmsg/checkbuffer.c)

Includes:

```
"sim_types.h"      "dgi_stdh.h"
"dgi_stdg.h"      "sim_cig_if.h"
"msg_loc.h"       "gbuffer.h"
"if_ctas.h"
```

2.1.2.2.2.18.1 check_buffer

This routine checks if a valid buffer is being sent to the CIG. It is called at each frame and verifies that the information in the buffer agrees with the information that is indicated in the message headers. It also checks to ensure that no invalid message types are being sent. If a problem with the buffer is found, an error message is printed. *mbp* is a pointer to the message buffer, and *bnum* is the buffer identifier.

Parameters		
Parameter	Type	Where Typedef Declared
mbp	pointer of MSGS_BLK	sim_cig_if.h
bnum	int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
bytes done	int	Standard
found end	int	Standard
hp	MSG HDR	sim_cig_if.h
error found	int	Standard
Return Values		
Return Value	Type	Meaning
error_found	BOOLEAN	TRUE - an error in the buffer was found FALSE - no error was found
Calls		
Function	Where Described	
DOT	gbuffer.h (macro definition)	
get ballistics debug	Sections 2.1.2.2.6.2 and 2.1.2.2.7.2	
print msg show effect	Section 2.1.2.2.2.56.1	
print msg file descr	Section 2.1.2.2.2.45.1	
print msg file xfer	Section 2.1.2.2.2.47.1	
print msg staticveh state	Section 2.1.2.2.2.58.1	
print msg staticveh rem	Section 2.1.2.2.2.57.1	
print msg process round	Section 2.1.2.2.2.54.1	
check all	Section 2.1.2.2.2.17.1	

Table 2.1-192: check_buffer Information.

2.1.2.2.2.18.2 print_msg_pass_on

This routine prints the first six elements of the Pass On message. It is used as a debugging tool. *hp* is a pointer to the Pass On message, and *len* is the length of the message.

Parameters		
Parameter	Type	Where Typedef Declared
hp	pointer ot MSG_PASS_ON	sim_cig_if.h
len	WORD	mass_std.h
Internal Variables		
Internal Variable	Type	Where Typedef Declared
i	int	Standard

Table 2.1-193: print_msg_pass_on Information.

2.1.2.2.2.19 cig_flushbuf.c

(./simnet/release/src/vehicle/libsrc/libmsg/cig_flushbuf.c)

Includes:

"stdio.h"

2.1.2.2.2.19.1 cig_flush_buffer

This routine flushes the CIG buffer.

Calls	
Function	Where Described
flush_buffer	Section 2.1.2.2.2.28.1

Table 2.1-194: cig_flush_buffer Information.

2.1.2.2.2.20 clr_n_mapped.c

(./simnet/release/src/vehicle/libsrc/libmsg/clr_n_mapped.c)

Includes:

"stdio.h"

"sim_types.h"

"mass_std.h"

"dgi_stdg.h"

"sim_cig_if.h"

"msg_loc.h"

2.1.2.2.2.20.1 clear_n_mapped

This routine sets *n_mapped* to 0.

2.1.2.2.2.21 config_key.c

(/simnet/release/src/vehicle/libsrc/libmsg/config_key.c)

Includes:

```
"stdio.h"
"sim_dfns.h"
"sim_types.h"
"sim_macros.h"
"mass_stdh.h"
"dgi_stdg.h"
"sim_cig_if.h"
"config_key.h"
```

The following static declarations are made:

```
cn
vs
tt
te
os
as
top_key_list[10]
node_key_list[12]
ball_key_list[10]
viewport_key_list[10]
```

2.1.2.2.2.21.1 key_list_initialized

This routine returns a flag, *list_inited*, which indicates whether or not the configuration tree was initialized.

Return Values		
Return Value	Type	Meaning
list_inited	int	TRUE = list was initialized FALSE = list not initialized

Table 2.1-195: key_list_initialized Information.

2.1.2.2.2.21.2 key_list_init

This routine initializes the configuration tree.

Internal Variables		
Internal Variable	Type	Where Typedef Declared
cnP	pointer to char	Standard
vsP	pointer to char	Standard
ttP	pointer to char	Standard
teP	pointer to char	Standard
osP	pointer to char	Standard
asP	pointer to char	Standard
Calls		
Function	Where Described	
add_keyword	Section 2.1.2.2.2.21.3	

Table 2.1-196: key_list_init Information.

2.1.2.2.2.21.3 add_keyword

This routine adds a node to the configuration tree. The parameters are defined as follows:

listP - pointer to list
keyword - pointer to a keyword
type - variable type
offset - offset
length - length
state - state

Parameters		
Parameter	Type	Where Typedef Declared
listP	pointer to KEY_LIST	Section 2.1.2.2.2.22 config_key.h
keyword	pointer to char	Standard
type	int	Standard
offset	int	Standard
length	int	Standard
state	int	Standard

Table 2.1-197: add_keyword Information.

2.1.2.2.2.21.4 lookup_keyword

This routine searches the keyword lists for the keyword. It returns a pointer to the list containing the keyword if it finds a match; otherwise, it returns NULL. The parameters are defined as follows:

keyword - keyword to be matched
table_list - starting list

Parameters		
Parameter	Type	Where Typedef Declared
keyword	pointer to char	Standard
table_list	pointer to pointer to KEY_LIST	Section 2.1.2.2.2.22 config_key.h
Internal Variables		
Internal Variable	Type	Where Typedef Declared
listP	pointer to KEY_LIST	Section 2.1.2.2.2.22 config_key.h
Return Values		
Return Value	Type	Meaning
NULL	pointer to KEY_LIST	Section 2.1.2.2.2.22 config_key.h
listP	pointer to KEY_LIST	keyword structure

Table 2.1-198: lookup_keyword Information.

2.1.2.2.2.22 config_key.h

(./simnet/release/src/vehicle/libsrc/libmsg/config_key.h)

The following constants are defined:

KEY_LIST_HDR
 MAX_KEYWORD_LENGTH
 MAX_INPUT_LENGTH
 TYPE_HWORD
 TYPE_STR
 TYPE_REAL
 TYPE_BYTE
 TYPE_WORD
 TYPE_INT_2
 TYPE_INT_4
 KEY_STATE_INIT
 KEY_STATE_NODE
 KEY_STATE_VIEWPORT
 KEY_STATE_TRAJ_TABLE
 KEY_STATE_TRAJ_ENTRY
 KEY_STATE_OVERLAY
 KEY_STATE_AGL

The KEY_LIST structure is defined.

The following are declared:

```
lookup_keyword()
key_list_init()
*top_key_listP
*node_key_listP
*viewport_key_listP
*ball_key_listP
*overlay_key_listP
*agl_key_listP
```

2.1.2.2.2.23 config_msg.c

(./simnet/release/src/vehicle/libsrc/libmsg/config_msg.c)

Includes:

```
"stdio.h"
"math.h"
"sim_dfns.h"
"sim_types.h"
"sim_macros.h"
"mass_std.h"
"dgi_std.h"
"sim_cig_if.h"
"libcig.h"
"librva.h"
"libmsg.h"
"cig_buffer.h"
"gbuffer.h"
"msg_loc.h"
```

The following are declared:

```
view_config_file[80]
traj_config_file[80]
```

2.1.2.2.2.23.1 cig_set_view_config_file

This routine sets the viewport configuration file name to the file name pointed to by *file_name*.

Parameters		
Parameter	Type	Where Typedef Declared
file_name	pointer to char	Standard

Table 2.1-199: cig_set_view_config_file Information.

2.1.2.2.2.23.2 cig_set_traj_config_file

This routine sets the trajectory configuration file name to the name pointed to by *file_name*.

Parameters		
Parameter	Type	Where Typedef Declared
file_name	pointer to char	Standard

Table 2.1-200: cig_set_traj_config_file Information.

2.1.2.2.2.23.3 cig_msg_configure_traj

This routine calls the routine `cig_read_config_file()` to read the trajectory configuration file.

Calls	
Function	Where Described
cig_read_configfile	Section 2.1.2.2.2.24.3

Table 2.1-201: cig_msg_configure_traj Information.

2.1.2.2.2.23.4 cig_msg_configure_view

This routine calls `flush_buffer()` to initialize the send buffer. It then calls `push_msg_cig_ctl()` to append the CIG Control message to the message buffer. The routine `cig_read_configfile()` is called; the viewport configuration file is read. The Viewflags message is appended to the CIG message buffer via `cig_msg_append_viewflags()`, and *buf_num* is reinitialized to 0. The parameter *buf_index* is the identifier for the send buffer.

Parameters		
Parameter	Type	Where Typedef Declared
buf_index	int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
other_buf	int	Standard
Calls		
Function	Where Described	
flush_buffer	Section 2.1.2.2.2.28.1	
push_msg_cig_ctl	Section 2.1.2.2.2.68.1	
cig_read_configfile	Section 2.1.2.2.2.24.3	
cig_msg_append_viewflags	Section 2.1.2.2.2.12.1	

Table 2.1-202: cig_msg_configure_view Information.

2.1.2.2.2.24 config_read.c

(.simnet/release/src/vehicle/libsrc/libmsg/config_read.c)

Includes:

```

"stdio.h"
"ctype.h"
"sim_stdio.h"
"sim_dfns.h"
"sim_types.h"
"sim_macros.h"
"mass_std.h"
"dgi_stdg.h"
"sim_cig_if.h"
"config_key.h"
"msg_loc.h"

```

The following are declared:

```

DT
init
*baseP

```

2.1.2.2.2.24.1 config_pos_init

This routine sets the initial position of the vehicle. The parameters are defined as follows:

pos - initial position vector.
head - initial heading, in radians.

Parameters		
Parameter	Type	Where Typedef Declared
<i>pos</i>	VECTOR	sim_types.h
<i>head</i>	REAL	sim_types.h
Internal Variables		
Internal Variable	Type	Where Typedef Declared
WORLD body to origin	VECTOR	sim_types.h
WORLD origin to body	VECTOR	sim_types.h
Calls		
Function	Where Described	
mat rot init	Section 2.6.2.47.1	
vec copy	Section 2.6.2.59.1	
vec neg	Section 2.6.2.62.1	
vec mat mul	Section 2.6.2.56.1	

Table 2.1-203: config_pos_init Information.

2.1.2.2.2.24.2 config_pos_init2

This routine sets the initial position of the vehicle. The parameters are defined as follows:

pos - initial position vector.
rot - initial rotation matrix.

Parameters		
Parameter	Type	Where Typedef Declared
<i>pos</i>	VECTOR	sim_types.h
<i>rot</i>	T_MATRIX	sim_types.h
Internal Variables		
Internal Variable	Type	Where Typedef Declared
WORLD body to origin	VECTOR	sim_types.h
WORLD origin to body	VECTOR	sim_types.h
Calls		
Function	Where Described	
mat_copy	Section 2.6.2.39.1	
vec_copy	Section 2.6.2.59.1	
vec_neg	Section 2.6.2.62.1	
vec_mat_mul	Section 2.6.2.56.1	

Table 2.1-204: config_pos_init2 Information.

2.1.2.2.2.24.3 cig_read_configfile

This routine reads the configuration file via `read_keyword_data()`, and processes the configuration file data via `process_keyword()`. The parameters are defined as follows:

file_name - the configuration file to be read
config_nameP - type of configuration file (view or trajectory)

Parameters		
Parameter	Type	Where Typedef Declared
<i>file_name</i>	pointer to char	Standard
<i>config_nameP</i>	pointer to char	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
<i>keywordP</i>	pointer to KEY_LIST	Section 2.1.2.2.2.22 config_key.h
<i>table_list[4]</i>	pointer to KEY_LIST	Section 2.1.2.2.2.22 config_key.h
<i>state</i>	int	Standard
<i>keyword[MAX_KEYWORD_LENGTH]</i>	char	Standard
<i>temp[MAX_INPUT_LENGTH]</i>	char	Standard
<i>lineno</i>	int	Standard
Calls		
Function	Where Described	
<i>key_list initialized</i>	Section 2.1.2.2.2.21.1	
<i>lookup keyword</i>	Section 2.1.2.2.2.21.4	
<i>read keyword data</i>	Section 2.1.2.2.2.24.4	
<i>copy to TF1</i>	Section 2.6.4.2.1	
<i>process keyword</i>	Section 2.1.2.2.2.24.5	

Table 2.1-205: cig_read_configfile Information.

2.1.2.2.2.24.4 read_keyword_data

This routine reads the contents of the configuration file. The parameters are defined as follows:

keywordP - pointer to key list
input_str - input line from file

Parameters		
Parameter	Type	Where Typedef Declared
keywordP	pointer to KEY_LIST	Section 2.1.2.2.2.22 config_key.h
input_str	pointer to char	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
num[10]	int	Standard
temp[100]	char	Standard

Table 2.1-206: read_keyword_data Information.

2.1.2.2.2.24.5 process_keyword

This routine processes the configuration file data.

Parameters		
Parameter	Type	Where Typedef Declared
state	int	Standard
keywordP	pointer to KEY_LIST	Section 2.1.2.2.2.22 config_key.h
table_list[]	pointer to KEY_LIST	Section 2.1.2.2.2.22 config_key.h
Return Values		
Return Value	Type	Meaning
keywordP->state	int	
Calls		
Function	Where Described	
append_msg_hdr	Section 2.1.2.2.2.5.1	
send_buffer	Section 2.1.2.2.2.24.6	

Table 2.1-207: process_keyword Information.

2.1.2.2.2.24.6 send_buffer

This routine sends the configuration buffer to the CIG. After the buffer has been sent, the Simulator Host waits for an acknowledgement that the buffer has been received by the CIG. After receiving a reply buffer from the CIG, the send buffer is prepared for the next message to be sent.

Calls	
Function	Where Described
copy_to_TF1	Section 2.6.4.2.1
cig_send_buffer	Section 2.1.2.2.1.10.1
cig_receive_buffer	Section 2.1.2.2.1.9.1
flush_buffer	Section 2.1.2.2.2.28.1

Table 2.1-208: send_buffer Information.

2.1.2.2.2.25 dealloc_abuf.c

(./simnet/release/src/vehicle/libsrc/libmsg/dealloc_abuf.c)

Includes:

"stdio.h"
 "sim_types.h"
 "mass_stdh.h"
 "dgi_stdg.h"
 "sim_cig_if.h"
 "msg_loc.h"
 "msg_loc.h"

2.1.2.2.2.25.1 deallocate_appended_buffer_space

This routine deallocates the most recently acquired appended buffer space. The size of the buffer space is denoted by *length*.

Parameters		
Parameter	Type	Where Typedef Declared
length	int	Standard

Table 2.1-209: deallocate_appended_buffer_space Information.

2.1.2.2.2.26 dealloc_pbuf.c

(./simnet/release/src/vehicle/libsrc/libmsg/dealloc_pbuf.c)

Includes:

"stdio.h"
"sim_types.h"
"mass_std.h"
"dgi_std.h"
sim_cig_if.h"
msg_loc.h"
"msg_loc.h"

2.1.2.2.2.26.1 deallocate_prepended_buffer_space

This routine deallocates the most recently acquired prependded buffer space. The size of the buffer space is denoted by *length*.

Parameters		
Parameter	Type	Where Typedef Declared
length	int	Standard

Table 2.1-210: deallocate_prepended_buffer_space Information.

2.1.2.2.2.27 del_veh.c

(./simnet/release/src/vehicle/libsrc/libmsg/del_veh.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_std.h"
"dgi_std.h"
"sim_cig_if.h"
"msg_loc.h"
"librva.h"
"gbuffer.h"
"cig_buffer.h"
```

2.1.2.2.2.27.1 delete_veh_from_cig_msg

This routine deletes a vehicle from the CIG message. *r_del* is a pointer to the RVA entry which indicates which vehicle is to be deleted.

Parameters		
Parameter	Type	Where Typedef Declared
r_del	pointer to RVA_ENTRY	librva.h
Internal Variables		
Internal Variable	Type	Where Typedef Declared
r_move	pointer to RVA_ENTRY	librva.h
Return Values		
Return Value	Type	Meaning
TRUE	int	vehicle removed
FALSE	int	vehicle not removed
Calls		
Function	Where Described	
REPORT_ERROR	gbuffer.h (macro definition)	

Table 2.1-211: delete_veh_from_cig_msg Information.

2.1.2.2.2.28 flushbuf.c

(./simnet/release/src/vehicle/libsrc/libmsg/flushbuf.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_stdh.h"
"dgi_stdh.h"
"sim_cig_if.h"
"msg_loc.h"
```

2.1.2.2.2.28.1 flush_buffer

This routine is used to initialize the send buffer for the next message.

2.1.2.2.2.29 get_back.c

(./simnet/release/src/vehicle/libsrc/libmsg/get_back.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_stdh.h"
"dgi_stdh.h"
"sim_cig_if.h"
"msg_loc.h"
```

2.1.2.2.2.29.1 get_back_of_send_buffer

This routine returns the location of the pointer to the back of the send buffer.

Parameters		
Parameter	Type	Where Typedef Declared
buf_index	int	Standard
Return Values		
Return Value	Type	Meaning
s_buffer_ptrs[buf_index] .back of send buffer	pointer to char	location of the pointer to the back of the send buffer

Table 2.1-212: get-back_of_send_buffer Information.

2.1.2.2.2.30 get_cig_mask.c

(./simnet/release/src/vehicle/libsrc/libmsg/get_cig_mask.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_std.h"
"dgi_std.h"
"sim_cig_if.h"
"msg_loc.h"
```

2.1.2.2.2.30.1 get_cig_mask

This routine returns the CIG mask. The CIG mask indicates which CIGs exist in the system.

Return Values		
Return Value	Type	Meaning
cig_mask	int	the CIGs which are present in the system

Table 2.1-213: get_cig_mask Information.

2.1.2.2.2.31 get_debug.c

(./simnet/release/src/vehicle/libsrc/libmsg/get_debug.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_std.h"
"dgi_std.h"
"sim_cig_if.h"
"msg_loc.h"
```

2.1.2.2.2.31.1 get_static_debug

This routine returns the value of *static_debug*, which indicates whether or not static vehicle debugging is enabled.

Return Values		
Return Value	Type	Meaning
static_debug	int	indicates whether static vehicle debugging is activated.

Table 2.1-214: get_static_debug Information.

2.1.2.2.2.32 get_front.c

(./simnet/release/src/vehicle/libsrc/libmsg/get_front.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_stdh.h"
"dgi_stdg.h"
"sim_cig_if.h"
"msg_loc.h"
```

2.1.2.2.2.32.1 get_front_of_send_buffer

This routine returns the location of the pointer to the front of the send buffer. *buf_index* indicates which CIG's send buffer is being accessed.

Parameters		
Parameter	Type	Where Typedef Declared
buf_index	int	Standard
Return Values		
Return Value	Type	Meaning
s_buffer_ptrs[buf_index] .front of send buffer	pointer to char	location of the pointer to the front of the send buffer

Table 2.1-215: get_front_of_send_buffer Information.

2.1.2.2.2.33 get_init_buf.c

(./simnet/release/src/vehicle/libsrc/libmsg/get_init_buf.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_stdh.h"
"dgi_stdg.h"
"sim_cig_if.h"
"msg_loc.h"
"libmsg.h"
```

2.1.2.2.2.33.1 get_init_ptrs

This routine returns the initial locations of the pointers for the CIG buffer.

Return Values		
Return Value	Type	Meaning
init_ptrs	pointer to S_3BUFFER	initial location of the pointers to the CIG buffer

Table 2.1-216: get_init_ptrs Information.

2.1.2.2.2.34 get_n_mapped.c

(./simnet/release/src/vehicle/libsrc/libmsg/get_n_mapped.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_std.h"
"dgi_std.h"
"sim_cig_if.h"
"msg_loc.h"
```

2.1.2.2.2.34.1 get_n_mapped

This routine returns the number of dynamic vehicles in the CIG buffer.

Return Values		
Return Value	Type	Meaning
n_mapped	int	number of dynamic vehicles in the CIG buffer

Table 2.1-217: get_n_mapped Information.

2.1.2.2.2.35 get_other_st.c

(./simnet/release/src/vehicle/libsrc/libmsg/get_other_st.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_std.h"
"dgi_std.h"
"sim_cig_if.h"
"msg_loc.h"
```

2.1.2.2.2.35.1 get_other_start_in_send_buffer

This routine returns the location of the pointer to the other dynamic vehicles.

Parameters		
Parameter	Type	Where Typedef Declared
buf_index	int	Standard
Return Values		
Return Value	Type	Meaning
s_buffer_ptrs[buf_index] .other start in send buffer	pointer to char	location of the pointer to the other dynamic vehicles

Table 2.1-218: get_other_start_in_send_buffer Information.

2.1.2.2.2.36 get_sbuffer.c

(./simnet/release/src/vehicle/libsrc/libmsg/get_sbuffer.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_std.h"
"dgi_std.h"
"sim_cig_if.h"
"msg_loc.h"
```

2.1.2.2.2.36.1 get_sbuffer

This routine returns the buffer which contains the initial locations of all of the CIG buffer pointers.

Return Values		
Return Value	Type	Meaning
s_buffer_ptrs	pointer to S_BUFFER	the buffer that contains the initial locations of all of the CIG buffer pointers

Table 2.1-219: get_sbuffer Information.

2.1.2.2.2.37 msg_loc.c

(./simnet/release/src/vehicle/libsrc/libmsg/msg_loc.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_std.h"
"dgi_std.h"
"sim_cig_if.h"
"gbuffer.h"
"librva.h"
"cig_buffer.h"
"libmsg.h"
"msg_loc.h"
```

This routine, given a message index into the slowly changing portion of the message buffer which holds descriptions of the visible remote mobile vehicles, indicates which remote_vehicles table it represents.

```
map_to_remote_vehicles[MAX_CLOSE_VEHICLES]
n_mapped
cig_msg_debug
```

The following are declared:

```
s_buffer_ptrs[2]      (so up to two CIGs can be supported)
init_ptrs[2]          (store values for pointer initial values)
buf_num
cig_mask
```

using_bumper_numbers
using_asyymmetric

The following pointers are used to keep track of trajectory chord messages and round fired messages which are stored in a temporary buffer until they can be copied into the CIG buffer.

*start_of_ballistics_buf	-start of allocated space
*front_of_ballistics_buf	-start of valid data
*end_of_ballistics_buf	-end of allocated space and valid data

2.1.2.2.2.38 msg_loc.h

(./simnet/release/src/vehicle/libsrc/libmsg/msg_loc.h)

Includes:

"sim_macros.h"
"cig_buffer.h"
"pro_sim.h"
"librva.h"
"libmsg.h"

The following are declared as external:

s_buffer_ptrs[]
cig_msg_debug
*map_to_remote_vehicles[MAX_CLOSE_VEHICLES]
n_mapped
static_debug
init_ptrs
buf_num
cig_mask
using_bumper_numbers
using_asyymmetric

The following external variables are pointers to the temporary buffer for ballistics messages:

*start_of_ballistics_buf
*front_of_ballistics_buf
*end_of_ballistics_buf

The following is defined:

MSG_LEN

The following macro is defined:

ptr_to_index(p)

2.1.2.2.2.40 pr_agl.c (./simnet/release/src/vehicle/libsrc/libmsg/pr_agl.c)

Includes:

```
"sim_types.h"
"mass_stdh.h"
"dgi_stdg.h"
"sim_cig_if.h"
```

2.1.2.2.2.40.1 print_msg_agl

This routine prints the AGL Collision message. It is used as a debugging tool. The parameters are defined as follows:

mp - a pointer to the message buffer.
length - the length of the message to be printed.

Parameters		
Parameter	Type	Where Typedef Declared
<i>mp</i>	pointer to MSG_AGL	sim_cig_if.h
<i>length</i>	int	Standard

Table 2.1-220: print_msg_agl Information.

2.1.2.2.2.41 pr_cig_ctl.c (./simnet/release/src/vehicle/libsrc/libmsg/pr_cig_ctl.c)

Includes:

```
"sim_types.h"
"mass_stdh.h"
"dgi_stdg.h"
"sim_cig_if.h"
```

2.1.2.2.2.41.1 print_msg_cig_ctl

This routine prints the CIG Control message. It is used as a debugging tool. The parameters are defined as follows:

mp - a pointer to the message buffer.
length - the length of the message to be printed.

Parameters		
Parameter	Type	Where Typedef Declared
<i>mp</i>	pointer to MSG_CIG_CTL	sim_cig_if.h
<i>length</i>	int	Standard

Table 2.1-221: print_msg_cig_ctl Information.

2.1.2.2.2.42 pr_ct_gm.c

(./simnet/release/src/vehicle/libsrc/libmsg/pr_ct_gm.c)

This file is not implemented in the Masscomp or Butterfly.

2.1.2.2.2.43 pr_ct_ism.c

(./simnet/release/src/vehicle/libsrc/libmsg/pr_ct_ism.c)

This file is not implemented in the Masscomp or Butterfly.

2.1.2.2.2.44 pr_end.c

(./simnet/release/src/vehicle/libsrc/libmsg/pr_end.c)

Includes:

```
"sim_types.h"
"mass_stdh.h"
"dgi_stdg.h"
"sim_cig_if.h"
```

2.1.2.2.2.44.1 print_msg_end

This routine prints an end message. It is used as a debugging tool.

2.1.2.2.2.45 pr_file_desc.c

(./simnet/release/src/vehicle/libsrc/libmsg/pr_file_desc.c)

Includes:

```
"sim_types.h"
"mass_stdh.h"
"dgi_stdg.h"
"sim_cig_if.h"
```

2.1.2.2.2.45.1 print_msg_file_descr

This routine prints a File Descriptor message. It is used as a debugging tool. The parameters are defined as follows:

mp - a pointer to the message buffer.
length - the length of the message to be printed.

Parameters		
Parameter	Type	Where Typedef Declared
<i>mp</i>	pointer to MSG_FILE_DESCR	sim_cig_if.h
<i>length</i>	int	Standard

Table 2.1-222: print_msg_file_descr Information.

2.1.2.2.2.46 pr_file_stat.c

(./simnet/release/src/vehicle/libsrc/libmsg/pr_file_stat.c)

Includes:

```
"sim_types.h"
"mass_std.h"
"dgi_stdg.h"
"sim_cig_if.h"
```

2.1.2.2.2.46.1 print_msg_file_status

This routine prints a File Status message. It is used as a debugging tool. The parameters are defined as follows:

mp - a pointer to the message buffer.
length - the length of the message to be printed.

Parameters		
Parameter	Type	Where Typedef Declared
<i>mp</i>	pointer to MSG_FILE_STATUS	sim_cig_if.h
<i>length</i>	int	Standard

Table 2.1-223: print_msg_file_status Information.

2.1.2.2.2.47 pr_file_xfer.c

(./simnet/release/src/vehicle/libsrc/libmsg/pr_file_xfer.c)

Includes:

```
"sim_types.h"
"mass_std.h"
"dgi_stdg.h"
"sim_cig_if.h"
```

2.1.2.2.2.47.1 print_msg_file_xfer

This routine prints a File Transfer message. It is used as a debugging tool.

The parameters are defined as follows:

mp - a pointer to the message buffer.
length - the length of the message to be printed.

Parameters		
Parameter	Type	Where Typedef Declared
<i>mp</i>	pointer to MSG_FILE_TRANSFER	sim_cig_if.h
<i>length</i>	int	Standard

Table 2.1-224: print_msg_file_xfer Information.

2.1.2.2.2.48 pr_hit.c

(./simnet/release/src/vehicle/libsrc/libmsg/pr_hit.c)

Includes:

"sim_types.h"
 "mass_std.h"
 "dgi_stdg.h"
 "sim_cig_if.h"

2.1.2.2.2.48.1 print_msg_hit

This routine prints a Hit message. It is used as a debugging tool. The parameters are defined as follows:

mp - a pointer to the message buffer.
length - the length of the message to be printed.

Parameters		
Parameter	Type	Where Typedef Declared
<i>mp</i>	pointer to MSG_HIT	sim_cig_if.h
<i>length</i>	int	Standard
Calls		
Function	Where Described	
print_R4P3D	Section 2.6.4.17.1	

Table 2.1-225: print_msg_hit Information.

2.1.2.2.2.49 pr_hit_rtn.c

(./simnet/release/src/vehicle/libsrc/libmsg/pr_hit_rtn.c)

Includes:

"sim_types.h"
 "mass_std.h"
 "dgi_stdg.h"
 "sim_cig_if.h"

2.1.2.2.2.49.1 print_msg_hit_return

This routine prints a Hit Return message. It is used as a debugging tool. The parameters are defined as follows:

mp - a pointer to the message buffer.
length - the length of the message to be printed.

Parameters		
Parameter	Type	Where Typedef Declared
<i>mp</i>	pointer to MSG_HIT_RETURN	sim_cig_if.h
<i>length</i>	int	Standard
Calls		
Function	Where Described	
print_R4P3D	Section 2.6.4.17.1	

Table 2.1-226: print_msg_hit_return Information.

2.1.2.2.2.50 pr_laser_rtn.c

(./simnet/release/src/vehicle/libsrc/libmsg/pr_laser_rtn.c)

Includes:

"sim_types.h"
 "mass_std.h"
 "dgi_stdg.h"
 "sim_cig_if.h"

2.1.2.2.2.50.1 print_msg_laser_return

This routine prints a Laser Return message. It is used as a debugging tool. The parameters are defined as follows:

mp - a pointer to the message buffer.
length - the length of the message to be printed.

Parameters		
Parameter	Type	Where Typedef Declared
<i>mp</i>	pointer to MSG_LASER_RETURN	sim_cig_if.h
<i>length</i>	int	Standard

Table 2.1-227: print_msg_laser_return Information.

2.1.2.2.2.51 pr_loc_terr.c

(./simnet/release/src/vehicle/libsrc/libmsg/pr_loc_terr.c)

Includes:

"sim_types.h"
 "mass_std.h"
 "dgi_stdg.h"
 "sim_cig_if.h"

2.1.2.2.2.51.1 print_msg_local_terrain

This routine prints a Local Terrain message. It is used as a debugging tool.

The parameters are defined as follows:

mp - a pointer to the message buffer.
length - the length of the message to be printed.

Parameters		
Parameter	Type	Where Typedef Declared
mp	pointer to MSG_LOCAL_TERRAIN	sim_cig_if.h
length	int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
i	int	Standard
v	int	Standard
poly	pointer to LT_POLY_ENTRY	sim_cig_if.h
bvol	pointer to LT_BVOL_ENTRY	sim_cig_if.h
foo	int	Standard
Calls		
Function	Where Described	
print_R4P3D	Section 2.6.4.17.1	

Table 2.1-228: print_msg_local_terrain Information.

2.1.2.2.2.52 pr_miss.c**(./simnet/release/src/vehicle/libsrc/libmsg/pr_miss.c)****Includes:**

"sim_types.h"
"mass_stdh.h"
"dgi_stdg.h"
"sim_cig_if.h"

2.1.2.2.2.52.1 print_msg_miss

This routine prints a Miss message. It is used as a debugging tool.

The parameters are defined as follows:

mp - a pointer to the message buffer.
length - the length of the message to be printed.

Parameters		
Parameter	Type	Where Typedef Declared
<i>mp</i>	pointer to MSG MISS	sim_cig_if.h
<i>length</i>	int	Standard

Table 2.1-229: print_msg_miss Information.

2.1.2.2.2.53 pr_otherveh.c

(./simnet/release/src/vehicle/libsrc/libmsg/pr_otherveh.c)

Includes:

"sim_types.h"
 "mass_std.h"
 "dgi_stdg.h"
 "sim_cig_if.h"

2.1.2.2.2.53.1 print_msg_otherveh_state

This routine prints an Other Vehicle State message. It is used as a debugging tool.

The parameters are defined as follows:

mp - a pointer to the message buffer.
length - the length of the message to be printed.

Parameters		
Parameter	Type	Where Typedef Declared
mp	pointer to MSG_OTHERVEH_STATE	sim_cig_if.h
length	int	Standard
Calls		
Function	Where Described	
print_TF1	Section 2.6.4.18.1	

Table 2.1-230: print_msg_otherveh_state Information.

2.1.2.2.2.54 pr_proc_rnd.c

(./simnet/release/src/vehicle/libsrc/libmsg/pr_proc_rnd.c)

Includes:

"sim_types.h"
 "mass_std.h"
 "dgi_stdg.h"
 "sim_cig_if.h"

2.1.2.2.2.54.1 print_msg_process_round

This routine prints a Process Round message. It is used as a debugging tool.

The parameters are defined as follows:

mp - a pointer to the message buffer.
length - the length of the message to be printed.

Parameters		
Parameter	Type	Where Typedef Declared
<i>mp</i>	pointer to MSG_PROCESS_ROUND	sim_cig_if.h
<i>length</i>	int	Standard
Calls		
Function	Where Described	
print_R4P3D	Section 2.6.4.17.1	

Table 2.1-231: print_msg_process_round Information.

2.1.2.2.2.55 pr_rnd_fired.c

(./simnet/release/src/vehicle/libsrc/libmsg/pr_rnd_fired.c)

Includes:

"sim_types.h"
"mass_std.h"
"dgi_stdg.h"
"sim_cig_if.h"

2.1.2.2.2.55.1 print_msg_round_fired

This routine prints a Round Fired message. It is used as a debugging tool.

The parameters are defined as follows:

mp - a pointer to the message buffer.
length - the length of the message to be printed.

Parameters		
Parameter	Type	Where Typedef Declared
mp	pointer to MSG_ROUND_FIRED	sim_cig_if.h
length	int	Standard
Calls		
Function	Where Described	
print_R4P3D	Section 2.6.4.17.1	

Table 2.1-232: print_msg_round_fired Information.

2.1.2.2.2.55 pr_rnd_fired.c

(./simnet/release/src/vehicle/libsrc/libmsg/pr_rnd_fired.c)

Includes:

"sim_types.h"
 "mass_std.h"
 "dgi_std.h"
 "sim_cig_if.h"

2.1.2.2.2.55.1 print_msg_round_fired

This routine prints a Round Fired message. It is used as a debugging tool.

The parameters are defined as follows:

mp - a pointer to the message buffer.
length - the length of the message to be printed.

Parameters		
Parameter	Type	Where Typedef Declared
mp	pointer to MSG_ROUND_FIRED	sim_cig_if.h
length	int	Standard
Calls		
Function	Where Described	
print_P4P3D	Section 2.6.4.17.1	

Table 2.1-232: print_msg_round_fired Information.

2.1.2.2.2.56 pr_show_eff.c

(/simnet/release/src/vehicle/libsrc/libmsg/pr_show_eff.c)

Includes:

```
"sim_types.h"
"mass_stdh.h"
"dgi_stdg.h"
"sim_cig_if.h"
```

2.1.2.2.2.56.1 print_msg_show_effect

This routine prints a Show Effect message. It is used as a debugging tool.

The parameters are defined as follows:

mp - a pointer to the message buffer.
length - the length of the message to be printed.

Parameters		
Parameter	Type	Where Typedef Declared
<i>mp</i>	pointer to MSG_SHOW_EFFECT	sim_cig_if.h
<i>length</i>	int	Standard
Calls		
Function	Where Described	
print_R4P3D	Section 2.6.4.17.1	

Table 2.1-233: print_msg_show_effect Information.

2.1.2.2.2.57 pr_staticrem.c**(./simnet/release/src/vehicle/libsrc/libmsg/pr_staticrem.c)**

Includes:

"sim_types.h"
"mass_stdh.h"
"dgi_stdg.h"
"sim_cig_if.h"

2.1.2.2.2.57.1 print_msg_staticveh_rem

This routine prints a Static Vehicle Remove message. It is used as a debugging tool.

The parameters are defined as follows:

mp - a pointer to the message buffer.
length - the length of the message to be printed.

Parameters		
Parameter	Type	Where Typedef Declared
<i>mp</i>	pointer to MSG_STATICVEH_REM	sim_cig_if.h
<i>length</i>	int	Standard

Table 2.1-234: print_msg_staticveh_rem Information.

2.1.2.2.2.58 pr_staticveh.c

(/simnet/release/src/vehicle/libsrc/libmsg/pr_staticveh.c)

Includes:

```
"sim_types.h"
"mass_stdh.h"
"dgi_stdg.h"
"sim_cig_if.h"
```

2.1.2.2.2.58.1 print_msg_staticveh_state

This routine prints a Static Vehicle State message. It is used as a debugging tool.

The parameters are defined as follows:

mp - a pointer to the message buffer.
length - the length of the message to be printed.

Parameters		
Parameter	Type	Where Typedef Declared
<i>mp</i>	pointer to MSG_STATICVEH_STATE	sim_cig_if.h
<i>length</i>	int	Standard
Calls		
Function	Where Described	
print TF1	Section 2.6.4.18.1	
print TF2	Section 2.6.4.19.1	

Table 2.1-235: print_msg_staticveh_state Information.

2.1.2.2.2.59 pr_submode.c

(./simnet/release/src/vehicle/libsrc/libmsg/pr_submode.c)

Includes:

```
"sim_types.h"
"mass_std.h"
"dgi_stdg.h"
"sim_cig_if.h"
```

2.1.2.2.2.59.1 print_msg_subsys_mode

This routine prints a Subsystem Mode message. It is used as a debugging tool.

The parameters are defined as follows:

mp - a pointer to the message buffer.
length - the length of the message to be printed.

Parameters		
Parameter	Type	Where Typedef Declared
<i>mp</i>	pointer to MSG_SUBSYS_MODE	sim_cig_if.h
<i>length</i>	int	Standard

Table 2.1-236: print_msg_subsys_mode Information.

2.1.2.2.2.60 pr_sys_err.c

(./simnet/release/src/vehicle/libsrc/libmsg/pr_sys_err.c)

Includes:

```
"sim_types.h"
"mass_std.h"
"dgi_stdg.h"
"sim_cig_if.h"
```

2.1.2.2.2.60.1 print_msg_sys_error

This routine prints a System Error message. It is used as a debugging tool.

The parameters are defined as follows:

mp - a pointer to the message buffer.
length - the length of the message to be printed.

Parameters		
Parameter	Type	Where Typedef Declared
<i>mp</i>	pointer to MSG_SYS_ERROR	sim_cig_if.h
<i>length</i>	int	Standard

Table 2.1-237: print_msg_sys_error Information.

2.1.2.2.2.61 pr_test_name.c

(./simnet/release/src/vehicle/libsrc/libmsg/pr_test_name.c)

Includes:

```
"sim_types.h"
"mass_std.h"
"dgi_stdg.h"
"sim_cig_if.h"
```

2.1.2.2.2.61.1 print_msg_test_name

This routine prints a Test Name message. It is used as a debugging tool.

The parameters are defined as follows:

mp - a pointer to the message buffer.
length - the length of the message to be printed.

Parameters		
Parameter	Type	Where Typedef Declared
<i>mp</i>	pointer to MSG TEST_NAME	sim_cig_if.h
<i>length</i>	int	Standard
Calls		
Function	Where Described	
print_R4P3D	Section 2.6.4.17.1	

Table 2.1-238: print_msg_test_name Information.

2.1.2.2.2.62 pr_traj_chrd.c

(./simnet/release/src/vehicle/libsrc/libmsg/pr_traj_chrd.c)

Includes:

"sim_types.h"
 "mass_stdh.h"
 "dgi_stdg.h"
 "sim_cig_if.h"

2.1.2.2.2.62.1 print_msg_traj_chord

This routine prints a Process Round message. It is used as a debugging tool.

The parameters are defined as follows:

mp - a pointer to the message buffer.
length - the length of the message to be printed.

Parameters		
Parameter	Type	Where Typedef Declared
mp	pointer to MSG_TRAJ_CHORD	sim_cig_if.h
length	int	Standard
Calls		
Function	Where Described	
print_R4P3D	Section 2.6.4.17.1	

Table 2.1-239: print_msg_traj_chord Information.

2.1.2.2.2.63 pr_vupdate.c

(./simnet/release/src/vehicle/libsrc/libmsg/pr_vupdate.c)

This file is not implemented in the Masscomp or the Butterfly.

2.1.2.2.2.64 pre_1rot.c

(./simnet/release/src/vehicle/libsrc/libmsg/pre_1rot.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_std.h"
"dgi_stdg.h"
"sim_cig_if.h"
"libmsg.h"
```

2.1.2.2.2.64.1 cig_msg_prepend_1rotation

This routine prepends the 1Rotation message to the CIG message buffer. The parameters represent information that will be added to the buffer, and are defined as follows:

- node_index* - the node that will be updated by the rotations in the following parameters.
- rotation_axis* - the axis of rotation, using one of the following values:
- 0 heading
 - 1 pitch
 - 2 roll
- rotation* - the rotation in degrees.

Parameters		
Parameter	Type	Where Typedef Declared
node_index	HWORD	mass_std.h
rotation_axis	BYTE	sim_types.h
rotation	REAL_4	mass_std.h
Internal Variables		
Internal Variable	Type	Where Typedef Declared
mp	register pointer to MSG 1ROTATION	sim_cig_if.h
Calls		
Function	Where Described	
prepend_msg_hdr	Section 2.1.2.2.2.83.1	

Table 2.1-240: cig_msg_prepend_1rotation Information.

2.1.2.2.2.65 pre_3rot.c

(./simnet/release/src/vehicle/libsrc/libmsg/pre_3rot.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_std.h"
"dgi_std.h"
"sim_cig_if.h"
"libmsg.h"
```

2.1.2.2.2.65.1 cig_msg_prepend_3rotations

This routine prepends the 3Rotation message to the CIG message buffer. The parameters represent information that will be added to the buffer, and are defined as follows:

node_index - the node that will be updated by the rotations in the following parameters.
heading - the heading, in degrees.
pitch - the pitch, in degrees.
roll - the roll in, degrees.

Parameters		
Parameter	Type	Where Typedef Declared
node_index	HWORD	mass_std.h
heading	REAL 4	mass_std.h
pitch	REAL 4	mass_std.h
roll	REAL 4	mass_std.h
Internal Variables		
Internal Variable	Type	Where Typedef Declared
mp	register pointer to MSG 3ROTATIONS	sim_cig_if.h
Calls		
Function	Where Described	
prepend_msg_hdr	Section 2.1.2.2.2.83.1	

Table 2.1-241: cig_msg_prepend_3rotations Information.

2.1.2.2.2.66 pre_agl_set.c

(./simnet/release/src/vehicle/libsrc/libmsg/pre_agl_set.c)

Includes:

"stdio.h"
 "sim_types.h"
 "mass_stdh.h"
 "dgi_stdh.h"
 "sim_cig_if.h"
 "libmsg.h"

2.1.2.2.2.66.1 cig_msg_prepend_agl_setup

This routine prepends the AGL Setup message to the CIG message buffer. The parameter *state* toggles between 0 and 1. A value of 0 indicates that the AGL should not be sent, and a value of 1 indicates that the AGL should be sent.

Parameters		
Parameter	Type	Where Typedef Declared
state	int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
mp	register pointer to MSG AGL SETUP	sim_cig_if.h
Calls		
Function	Where Described	
prepend_msg_hdr	Section 2.1.2.2.2.83.1	

Table 2.1-242: cig_msg_prepend_agl_setup Information.

2.1.2.2.2.67 pre_am_dfn.c

(/simnet/release/src/vehicle/libsrc/libmsg/pre_am_dfn.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_std.h"
"dgi_std.h"
"sim_cig_if.h"
"libmsg.h"
```

2.1.2.2.2.67.1 cig_msg_prepend_ammo_define

This routine prepends the Ammo Define message to the CIG message buffer. The parameters represent information that will be added to the buffer, and are defined as follows:

```
ammo0      - ammunition type.
ammo1      - ammunition type.
ammo2      - ammunition type.
ammo3      - ammunition type.
```

Parameters		
Parameter	Type	Where Typedef Declared
ammo0	inr	Standard
ammo1	int	Standard
ammo2	int	Standard
ammo3	int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
mp	register pointer to MSG AMMO DEFINE	sim_cig_if.h
Calls		
Function	Where Described	
prepend_msg_hdr	Section 2.1.2.2.2.83.1	

Table 2.1-243: cig_msg_prepend_aammo_define Information.

2.1.2.2.2.68 pre_cig_ctl.c

(./simnet/release/src/vehicle/libsrc/libmsg/pre_cig_ctl.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_std.h"
"dgi_std.h"
"sim_cig_if.h"
"libmsg.h"
```

2.1.2.2.2.68.1 cig_msg_prepend_cig_ctl

This routine prepends the CIG Control message to the CIG message buffer. The parameter *state* is a CIG control code which corresponds to the following:

- 0 Do Nothing (NULL)
- 1 Prepare for Simulation
- 2 Run Simulation Mode
- 5 File Transfer Mode
- 6 Run Tests
- 7 Stop
- 8 Start CIG Configuration
- 9 Start 2-D Setup

Parameters		
Parameter	Type	Where Typedef Declared
state	int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
mp	register pointer to MSG CIG CTL	sim_cig_if.h
Calls		
Function	Where Described	
prepend_msg_hdr	Section 2.1.2.2.2.83.1	

Table 2.1-244: cig_msg_prepend_cig_ctl Information.

2.1.2.2.2.69 pre_ct_gm.c

(./simnet/release/src/vehicle/libsrc/libmsg/pre_ct_gm.c)

This file is not implemented in the Masscomp or the Butterfly.

2.1.2.2.2.70 pre_ct_ism.c

(./simnet/release/src/vehicle/libsrc/libmsg/pre_ct_ism.c)

This file is not implemented in the Masscomp or the Butterfly.

2.1.2.2.2.71 pre_dr11.c

(./simnet/release/src/vehicle/libsrc/libmsg/pre_dr11.c)

Includes:

"stdio.h"
 "sim_types.h"
 "mass_std.h"
 "dgi_std.h"
 "sim_cig_if.h"
 "libmsg.h"

2.1.2.2.2.71.1 cig_msg_prepend_dr11_pkt_size

This routine prepends the DR11 Packet Size message to the CIG message buffer. The parameters represent information that will be added to the buffer, and are defined as follows:

- send_size* - size of the DR11-W packet that the simulation host sends to the CIG at each frame.
- recv_size* - size of the DR11-W packet that the CIG sends to the simulation host at each frame.
- lt_chunk_size* - size of the local terrain pieces that are sent from the CIG to the simulation host until the entire local terrain has been sent. The value must be less than the CIG packet size by at least 50 bytes.
- lt_interval* - the interval, in frames, between transmissions of local terrain messages.
- hw_type* - the hardware type of the CIG.

Parameters		
Parameter	Type	Where Typedef Declared
send_size	int	Standard
recv_size	int	Standard
lt_chunk_size	int	Standard
lt_interval	int	Standard
hw_type	int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
mp	register pointer to MSG_DR11_PKT_SIZE	sim_cig_if.h
Calls		
Function	Where Described	
prepend_msg_hdr	Section 2.1.2.2.83.1	

Table 2.1-245: cig_msg_prepend_dr11_pkt_size Information.

2.1.2.2.2.72 pre_file_des.c

(./simnet/release/src/vehicle/libsrc/libmsg/pre_file_des.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_std.h"
"dgi_stdg.h"
"sim_cig_if.h"
"libmsg.h"
```

2.1.2.2.2.72.1 push_msg_file_descr

This routine prepends the File Descriptor message to the CIG message buffer. The parameters represent information that will be added to the buffer, and are defined as follows:

<i>db_size</i>	- the size of the file to be transferred, in bytes.
<i>db_no</i>	- the number that has been assigned to the file to be downloaded or uploaded.
<i>db_req</i>	- specifies the requested database operation.
<i>db_name</i>	- the name of the file or database.

Parameters		
Parameter	Type	Where Typedef Declared
db_size	int	Standard
db_no	int	Standard
db_req	int	Standard
db_name	pointer to char	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
mp	register pointer to MSG_FILE_DESCR	sim_cig_if.h
Calls		
Function	Where Described	
prepend_msg_hdr	Section 2.1.2.2.83.1	

Table 2.1-246: push_msg_file_descr Information.

2.1.2.2.2.73 pre_file_sts.c

(./simnet/release/src/vehicle/libsrc/libmsg/pre_file_sts.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_stdc.h"
"dgi_stdg.h"
"sim_cig_if.h"
"libmsg.h"
```

2.1.2.2.2.73.1 push_msg_file_status

This routine prepends the File Status message to the CIG message buffer. The parameters represent information that will be added to the buffer, and are defined as follows:

type - the type of response specified.
abort_descr_no - the abort code.
blk_seq_no - the block response number.

Parameters		
Parameter	Type	Where Typedef Declared
<i>type</i>	int	Standard
<i>abort_descr_no</i>	int	Standard
<i>blk_seq_no</i>	int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
<i>mp</i>	register pointer to MSG FILE STATUS	sim_cig_if.h
Calls		
Function	Where Described	
prepend_msg_hdr	Section 2.1.2.2.2.83.1	

Table 2.1-247: push_msg_file_status Information.

2.1.2.2.2.74 pre_file_xfr.c

(./simmet/release/src/vehicle/libsrc/libmsg/pre_file_xfr.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_std.h"
"dgi_stdg.h"
"sim_cig_if.h"
"libmsg.h"
"msg_loc.h"
```

2.1.2.2.2.74.1 push_msg_file_xfer

This routine prepends the File Transfer message to the CIG message buffer. The parameters represent information that will be added to the buffer and are defined as follows:

type - the type of transfer that the block represents. 1 = file block, 2 = end of file.

blk_size - the amount, in bytes, of data in this block.

blk_seq_no - the block's sequence number in the file transfer process.

data - the data in the file.

Parameters		
Parameter	Type	Where Typedef Declared
<i>type</i>	int	Standard
<i>blk_size</i>	int	Standard
<i>blk_seq_no</i>	int	Standard
<i>data</i>	pointer to char	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
<i>len</i>	int	Standard
<i>mp</i>	register pointer to MSG FILE XFER	sim_cig_if.h
Calls		
Function	Where Described	
prepend msg_hdr	Section 2.1.2.2.2.83.1	

Table 2.1-248: push_msg_file_xfer Information.

2.1.2.2.2.75 pre_gun_over.c

(.simnet/release/src/vehicle/libsrc/libmsg/pre_gun_over.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_std.h"
"dgi_stdg.h"
"sim_cig_if.h"
"libmsg.h"
```

2.1.2.2.2.75.1 cig_msg_prepend_gun_overlay

This routine prepends the Gun Overlay message to the CIG message buffer. The parameters represent information that will be added to the buffer, and are defined as follows:

<i>type</i>	- type of vehicle
<i>lrf_rdy</i>	- the ready indicator
<i>lrf_mal</i>	- the malfunction indicator
<i>lrf_mrb</i>	- the multiple return indicator
<i>ammo</i>	- ammunition type
<i>lrf_rng[]</i>	- laser range value
<i>azimuth[]</i>	- the azimuth digits
<i>range[]</i>	- the range digits
<i>t_v_mtx</i>	- gun barrel rotation

Parameters		
Parameter	Type	Where Typedef Declared
<i>type</i>	int	Standard
<i>lrf_rdy</i>	int	Standard
<i>lrf_mal</i>	int	Standard
<i>lrf_mrb</i>	int	Standard
<i>ammo</i>	int	Standard
<i>lrf_rng[]</i>	BYTE	sim_types.h
<i>azimuth[]</i>	BYTE	sim_types.h
<i>range[]</i>	BYTE	sim_types.h
<i>t_v_mtx</i>	pointer to TF2	sim_cig_if.h
Internal Variables		
Internal Variable	Type	Where Typedef Declared
<i>mp</i>	register pointer to MSG_GUN_OVERLAY	sim_cig_if.h
Calls		
Function	Where Described	
copy TF2	Section 2.6.4.6.1	
prepend msg_hdr	Section 2.1.2.2.2.83.1	

Table 2.1-249: cig_msg_prepend_gun_overlay Information.

2.1.2.2.2.76 pre_hprxyzs.c

(/simnet/release/src/vehicle/libsrc/libmsg/pre_hprxyzs.c)

Includes:

```

"stdio.h"
"sim_types.h"
"mass_std.h"
"dgi_std.h"
"sim_cig_if.h"
"libmsg.h"
"msg_loc.h"

```

2.1.2.2.2.76.1 cig_msg_prepend_hprxyzs_matrix

This routine prepends the HPRXYZS Matrix message to the CIG message buffer. The parameters represent information that will be added to the buffer, and are defined as follows:

<i>node_index</i>	- the node that will be updated by the values in the following parameters
<i>heading</i>	- the heading, in degrees
<i>pitch</i>	- the pitch, in degrees
<i>roll</i>	- the roll, in degrees
<i>translation</i>	- the negative X, Y, and Z position, in meters
<i>scale</i>	- reserved for future expansion
<i>concat_order[]</i>	- order in which the matrices are multiplied

Parameters		
Parameter	Type	Where Typedef Declared
<i>node_index</i>	WORD	mass_std.h
<i>heading</i>	REAL 4	mass_std.h
<i>pitch</i>	REAL 4	mass_std.h
<i>roll</i>	REAL 4	mass_std.h
<i>translation</i>	pointer to R4P3D	dgi_std.h
<i>scale</i>	pointer to R4P3D	dgi_std.h
<i>concat_order[]</i>	BYTE	sim_types.h
Internal Variables		
Internal Variable	Type	Where Typedef Declared
<i>mp</i>	register pointer to MSG HPRXYZS MATRIX	sim_cig_if.h
Calls		
Function	Where Described	
<i>prepend_msg_hdr</i>	Section 2.1.2.2.2.83.1	

Table 2.1-250: **cig_msg_prepend_hprxyzs_matrix** Information.

2.1.2.2.2.77 pre_lase_rtn.c

(/simnet/release/src/vehicle/libsrc/libmsg/pre_lase_rtn.c)

Includes:

```

"stdio.h"
"sim_types.h"
"mass_std.h"
"dgi_std.h"
"sim_cig_if.h"
"libmsg.h"
"msg_loc.h"

```

2.1.2.2.2.77.1 push_msg_laser_return

This routine prepends the Laser Return message to the CIG message buffer. The parameter *z_range* represents the range of the object (in meters) in the indicated pixel. A value of the last valid return value should be specified if the object in the indicated pixel is "sky".

Parameters		
Parameter	Type	Where Typedef Declared
z_range	float	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
mp	register pointer to MSG LASER RETURN	sim_cig_if.h
Calls		
Function	Where Described	
prepend_msg_hdr	Section 2.1.2.2.2.83.1	

Table 2.1-251: push_msg_laser_return Information.

2.1.2.2.2.78 pre_mcig_ctl.c

(./simnet/release/src/vehicle/libsrc/libmsg/pre_mcig_ctl.c)

Includes:

"stdio.h"
 "sim_types.h"
 "mass_stdc.h"
 "dgi_stdg.h"
 "sim_cig_if.h"
 "libmsg.h"

2.1.2.2.2.78.1 multi_cig_push_cig_ctl

This routine prepends the CIG Control message to the appropriate CIG message buffer. The parameters represent information that will be added to the buffer, and are defined as follows:

buf_mask - indicates CIG buffer
state - a CIG control code which corresponds to the following:

0	Do Nothing (NULL)
1	Prepare for Simulation
2	Run Simulation Mode
5	File Transfer Mode
6	Run Tests
7	Stop
8	Start CIG Configuration
9	Start 2-D Setup

Parameters		
Parameter	Type	Where Typedef Declared
buf_mask	int	Standard
state	int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
mp	register pointer to MSG CIG CTL	sim_cig_if.h
Calls		
Function	Where Described	
cig_push_msg_ctl	Section 2.1.2.2.2.68.1	

Table 2.1-252: multi_cig_push_cig_ctl Information.

2.1.2.2.2.79 pre_mdr11.c

(./simnet/release/src/vehicle/libsrc/libmsg/pre_mdr11.c)

Includes:

"stdio.h"
 "sim_types.h"
 "mass_stdh.h"
 "dgi_stdh.h"
 "sim_cig_if.h"
 "libmsg.h"

2.1.2.2.2.79.1 multi_cig_msg_prepend_dr11_pkt_size

This routine prepends the DR11 Packet Size message to the appropriate CIG message buffer. The parameters represent information that will be added to the buffer, and are defined as follows:

buf_mask - indicates CIG buffer
send_size - size of the DR11-W packet that the simulation host sends to the CIG at each frame
recv_size - size of the DR11-W packet that the CIG sends to the simulation host at each frame
lt_chunk_size - size of the local terrain pieces that are sent from the CIG to the simulation host until the entire local terrain has been sent. The value must be less than the CIG packet size by at least 50 bytes.
lt_interval - the interval, in frames, between transmissions of local terrain messages
hw_type - the hardware type of the CIG

Parameters		
Parameter	Type	Where Typedef Declared
<i>buf_mask</i>	int	Standard
<i>send_size</i>	int	Standard
<i>recv_size</i>	int	Standard
<i>lt_chunk_size</i>	int	Standard
<i>lt_interval</i>	int	Standard
<i>hw_type</i>	int	Standard
Calls		
Function	Where Described	
<i>cig_msg_prepend_dr11_pkt_size</i>	Section 2.1.2.2.2.71.1	

Table 2.1-253: multi_cig_msg_prepend_dr11_pkt_size Information.

2.1.2.2.2.80 pre_mpass_on.c

(./simnet/release/src/vehicle/libsrc/libmsg/pre_mpass_on.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_std.h"
"dgi_std.h"
"sim_cig_if.h"
"libmsg.h"
"msg_loc.h"
```

2.1.2.2.2.80.1 multi_cig_msg_prepend_pass_on

This routine prepends the Pass On message to the appropriate CIG message buffer. The parameters represent information that will be added to the buffer, and are defined as follows:

<i>buf_mask</i>	- indicates CIG buffer
<i>subsys_id</i>	- the id of the subsystem where the data is to be sent
<i>subsys_msg</i>	- the message to be sent
<i>msg_length</i>	- the length of the message

Parameters		
Parameter	Type	Where Typedef Declared
buf_mask	int	Standard
subsys_id	int	Standard
subsys_msg	HWORD	mass_std.h
msg_length	int	Standard
Calls		
Function	Where Described	
cig_msg_prepend_pass_on	Section 2.1.2.2.2.89.1	

Table 2.1-254: multi_cig_msg_prepend_pass_on Information.

2.1.2.2.2.81 pre_mreq_1sr.c

(/simnet/release/src/vehicle/libsrc/libmsg/pre_mreq_1sr.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_stdh.h"
"dgi_stdg.h"
"sim_cig_if.h"
"libmsg.h"
"msg_loc.h"
```

2.1.2.2.2.81.1 multi_cig_msg_prepend_request_laser_range

This routine prepends the Request Laser Range message to the appropriate CIG message buffer. The parameters represent information that will be added to the buffer, and are defined as follows:

buf_mask - indicates CIG buffer
i - pixel locations within the screen boundry
j - pixel locations within the screen boundry
id - the id of the laser request. This value is returned with the laser range.

Parameters		
Parameter	Type	Where Typedef Declared
<i>buf_mask</i>	int	Standard
<i>i</i>	int	Standard
<i>j</i>	int	Standard
<i>id</i>	int	Standard
Calls		
Function	Where Described	
<i>cig_msg_prepend_request_laser_range</i>	Section 2.1.2.2.2.91.1	

Table 2.1-255: multi_cig_msg_prepend_request_laser_range Information.

2.1.2.2.2.82 pre_mrts4x3.c

(/simnet/release/src/vehicle/libsrc/libmsg/pre_mrts4x3.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_std.h"
"dgi_stdg.h"
"sim_cig_if.h"
"libmsg.h"
"msg_loc.h"
```

2.1.2.2.2.82.1 multi_cig_msg_prepend_rts4x3_matrix

This routine prepends the RTS4X3 Matrix Message to the appropriate CIG message buffer. The parameters represent information that will be added to the buffer, and are defined as follows:

<i>buf_mask</i>	- indicates CIG buffer.
<i>node_index</i>	- the node that will be updated by the 4x3 matrix.
<i>rot_mtx</i>	- the transformation matrix that describes the location of the vehicle.
<i>vec</i>	- X, Y, Z translation values that describe the location of the vehicle.

Parameters		
Parameter	Type	Where Typedef Declared
<i>buf_mask</i>	int	Standard
<i>node_index</i>	int	Standard
<i>rot_mtx</i>	T MATRIX	sim_types.h
<i>vec</i>	VECTOR	sim_types.h
Calls		
Function	Where Described	
<i>cig_msg_prepend_rts4x3_matrix</i>	Section 2.1.2.2.2.94.1	

Table 2.1-256: multi_cig_msg_prepend_rts4x3_matrix Information.

2.1.2.2.2.83 pre_msg_hdr.c

(./simnet/release/src/vehicle/libsrc/libmsg/pre_msg_hdr.c)

Includes:

```
"stdio.h"
"sim_dfns.h"
"sim_macros.h"
"sim_types.h"
"mass_stdh.h"
"dgi_stdg.h"
"sim_cig_if.h"
"gbuffer.h"
"msg_loc.h"
"libmsg.h"
```

2.1.2.2.2.83.1 prepend_msg_hdr

This routine is called to request the next piece of memory on the top of the current buffer. If there is enough memory available, it allocates *length* bytes of space plus a MSG_HDR which will be formatted to specify the passed type of the message. If unsuccessful, the routine returns a NULL pointer; otherwise, it returns a pointer to the memory that the caller should use for the message.

This routine checks two pointers to determine if there is still enough of the buffer available. It first checks the top of the total buffer allocated (start_of_send_buffer) leaving enough room for the MSGS_BLK header. It then checks against the current bottom of the send buffer (back_of_send_buffer), leaving room for both the MSGS_BLK header and for the MSG_HDR that will be used for the MSG_END.

Parameters		
Parameter	Type	Where Typedef Declared
type	int	Standard
length	int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
hp	register pointer to MSG_HDR	sim_cig_if.h
Return Values		
Return Value	Type	Meaning
NULL	pointer to char	invalid message type; not enough space
s_buffer_ptrs[buf_num].front_of_send_buffer+sizeof(MSG_HDR)	pointer to char	a pointer to space to be used by caller
Calls		
Function	Where Described	
get send size	Section 2.1.2.2.1.22.1	

Table 2.1-257: prepend_msg_hdr Information.

2.1.2.2.2.84 pre_mvflags.c

(/simnet/release/src/vehicle/libsrc/libmsg/pre_mvflags.c)

Includes:

"stdio.h"
 "sim_types.h"
 "mass_std.h"
 "dgi_std.h"
 "sim_cig_if.h"
 "libmsg.h"
 "msg_loc.h"

2.1.2.2.2.84.1 multi_cig_msg_prepend_view_flags

This routine prepends the View Flags message to the appropriate CIG message buffer. The parameters represent information that will be added to the buffer, and are defined as follows:

- buf_mask* - indicates CIG message buffer
- view_flags* - values that turn on or off individual processing paths in the CIG system
- branch_values* - an array of values that control the branching of the branch nodes. Configuration branch nodes use values from this array to determine which branch to take. These values are compared against the branch mask specified in the Create Configuration Node message.

Parameters		
Parameter	Type	Where Typedef Declared
<i>buf_mask</i>	int	Standard
<i>view_flags</i>	WORD	mass_std.h
<i>branch_value[]</i>	WORD	mass_std.h
Calls		
Function	Where Described	
cig_msg_prepend_view_flags	Section 2.1.2.2.2.106.1	

Table 2.1-258: multi_cig_msg_prepend_view_flags Information.

2.1.2.2.2.85 pre_obscure.c

(/simnet/release/src/vehicle/libsrc/libmsg/pre_obscure.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_std.h"
"dgi_std.h"
"sim_cig_if.h"
"libmsg.h"
```

2.1.2.2.2.85.1 cig_msg_prepend_obscure

This routine prepends the Obscure message to the CIG message buffer. The parameters represent information that will be added to the buffer, and are defined as follows:

channel - the viewport id to which the effect will be applied.
texture - the texture to be applied.
repeat - the number of times to repeat the texture.
glare - the amount of glare; values range from 0 to 15.

Parameters		
Parameter	Type	Where Typedef Declared
channel	inr	Standard
texture	int	Standard
repeat	int	Standard
glare	int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
mp	register pointer to MSG OBSCURE	sim_cig_if.h
Calls		
Function	Where Described	
prepend msg_hdr	Section 2.1.2.2.2.83.1	

Table 2.1-259: cig_msg_prepend_obscure Information.

2.1.2.2.2.86 pre_overall.c

(./simnet/release/src/vehicle/libsrc/libmsg/pre_overall.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_std.h"
"dgi_stdg.h"
"sim_cig_if.h"
"gbuffer.h"
"msg_loc.h"
```

2.1.2.2.2.86.1 cig_msg_prepend_overall_hdr

This routine prepends the Overall Header message to the CIG message buffer. *buf_index* is the buffer to which to write.

Parameters		
Parameter	Type	Where Typedef Declared
buf_index	int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
mbp	register pointer to MSGS_BLK	sim_cig_if.h
Calls		
Function	Where Described	
get_send_size	Section 2.1.2.2.1.22.1	
REPORT_ERROR	gbuffer.h (macro definition)	

Table 2.1-260: cig_msg_prepend_overall_hdr Information.

2.1.2.2.2.87 pre_ovr_set.c

(./simnet/release/src/vehicle/libsrc/libmsg/pre_ovr_set.c)

Includes:

"stdio.h"
 "sim_types.h"
 "mass_stdc.h"
 "dgi_stdg.h"
 "sim_cig_if.h"
 "libmsg.h"

2.1.2.2.2.87.1 cig_msg_prepend_overlay_setup

This routine prepends the Overlay Setup message to the CIG message buffer. The parameters represent information that will be added to the buffer, and are defined as follows:

type - type of overlay (M1 or M2)
node - node id.
offset - offset for gun barrel overlay

Parameters		
Parameter	Type	Where Typedef Declared
<i>type</i>	int	Standard
<i>node</i>	int	Standard
<i>offset</i>	VECTOR	sim_types.h
Internal Variables		
Internal Variable	Type	Where Typedef Declared
<i>mp</i>	register pointer to MSG OVERLAY SETUP	sim_cig_if.h
Calls		
Function	Where Described	
<i>prepend_msg_hdr</i>	Section 2.1.2.2.2.83.1	

Table 2.1-261: cig_msg_prepend_overlay_setup Information.

2.1.2.2.2.88 pre_pass_bk.c

(./simnet/release/src/vehicle/libsrc/libmsg/pre_pass_bk.c)

Includes:

"stdio.h"
 "sim_types.h"
 "mass_std.h"
 "dgi_std.h"
 "sim_cig_if.h"
 "libmsg.h"
 "msg_loc.h"

2.1.2.2.2.88.1 cig_msg_prepend_pass_bk

This routine prepends the Pass Back message to the CIG message buffer. The parameters represent information that will be added to the buffer, and are defined as follows:

subsys_id - the id of the subsystem where the data is to be sent.
subsys_msg - the message to be sent.
msg_length - the length of the message.

Parameters		
Parameter	Type	Where Typedef Declared
subsys id	INT 2	mass_std.h
subsys msg[]	BYTE	sim_types.h
msg length	INT 2	mass_std.h
Internal Variables		
Internal Variable	Type	Where Typedef Declared
mp	register pointer to MSG PASS BACK	sim_cig_if.h
Calls		
Function	Where Described	
prepend_msg_hdr	Section 2.1.2.2.83.1	

Table 2.1-262: cig_msg_prepend_pass_bk Information.

2.1.2.2.2.89 pre_pass_on.c

(/simnet/release/src/vehicle/libsrc/libmsg/pre_pass_on.c)

Includes:

```

"stdio.h"
"sim_types.h"
"mass_std.h"
"dgi_std.h"
"sim_cig_if.h"
"libmsg.h"
"msg_loc.h"

```

2.1.2.2.2.89.1 cig_msg_prepend_pass_on

This routine prepends the Pass On message to the CIG message buffer. The parameters represent information that will be added to the buffer, and are defined as follows:

subsys_id - the id of the subsystem where the data is to be sent.
subsys_msg - the message to be sent.
msg_length - the length of the message.

Parameters		
Parameter	Type	Where Typedef Declared
subsys_id	int	Standard
subsys_msg[]	HWND	mass_std.h
msg_length	int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
mp	register pointer to MSG PASS ON	sim_cig_if.h
Calls		
Function	Where Described	
prepend_msg_hdr	Section 2.1.2.2.83.1	

Table 2.1-263: cig_msg_prepend_pass_on Information.

2.1.2.2.2.90 pre_proc_rnd.c

(/simnet/release/src/vehicle/libsrc/libmsg/pre_proc_rnd.c)

Includes:

```

"stdio.h"
"sim_types.h"
"sim_dfns.h"
"mass_std.h"
"dgi_std.h"
"sim_cig_if.h"
"libmsg.h"

```

2.1.2.2.2.90.1 cig_msg_prepend_ballistics_msg

This routine prepends the Process Round message to the CIG message buffer. The parameters represent information that will be added to the buffer, and are defined as follows:

<i>type</i>	- round type.
<i>tracer</i>	- tracer effect. A value of 0 will cause the effect type in the trajectory table specified by round type to be displayed as the tracer; otherwise, a valid tracer effect must be specified.
<i>id</i>	- unique round identifier.
<i>gunpos</i>	- the gun position, in world coordinates, at the time of firing.
<i>gunvel</i>	- the gun elevation, in world coordinates, at the time of firing.
<i>siinelv</i>	- sine of the gun elevation angle, in world coordinates, at the time of firing.
<i>coselv</i>	- cosine of the gun elevation angle, in world coordinates, at the time of firing.
<i>sinazm</i>	- sine of the gun azimuth angle, in world coordinates, at the time of firing.
<i>cosazm</i>	- cosine of the gun azimuth angle, in world coordinates, at the time of firing.
<i>est_impact_time</i>	- estimated impact time.
<i>est_impact_range</i>	- estimated impact range.

Parameters		
Parameter	Type	Where Typedef Declared
<i>type</i>	int	Standard
<i>tracer</i>	int	Standard
<i>id</i>	int	Standard
<i>gunpos</i>	VECTOR	sim_types.h
<i>gunvel</i>	VECTOR	sim_types.h
<i>siinelv</i>	REAL	sim_types.h
<i>coselv</i>	REAL	sim_types.h
<i>sinazm</i>	REAL	sim_types.h
<i>cosazm</i>	REAL	sim_types.h
<i>est_impact_time</i>	REAL	sim_types.h
<i>est_impact_range</i>	REAL	sim_types.h
Internal Variables		
Internal Variable	Type	Where Typedef Declared
<i>mp</i>	register pointer to MSG_PROCESS_ROUND	sim_cig_if.h
Calls		
Function	Where Described	
<i>prepend_msg_hdr</i>	Section 2.1.2.2.2.83.1	

Table 2.1-264: cig_msg_prepend_ballistics_msg Information.

2.1.2.2.2.91 pre_req_1sr.c

(./simnet/release/src/vehicle/libsrc/libmsg/pre_req_1sr.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_stdh.h"
"dgi_stdh.h"
"sim_cig_if.h"
"libmsg.h"
```

2.1.2.2.2.91.1 cig_msg_prepend_request_laser_range

This routine prepends the Request Laser Range message to the CIG message buffer. The parameters represent information that will be added to the buffer, and are defined as follows:

- i* - pixel locations within the screen boundry.
- j* - pixel locations within the screen boundry.
- id* - the id of the laser request. This value is returned with the laser range.

Parameters		
Parameter	Type	Where Typedef Declared
<i>i</i>	int	Standard
<i>j</i>	int	Standard
<i>id</i>	int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
mp	register pointer to MSG_REQUEST_LASER_RANGE	sim_cig_if.h
Calls		
Function	Where Described	
prepend msg_hdr	Section 2.1.2.2.2.83.1	

Table 2.1-265: cig_msg_prepend_request_laser_range Information.

2.1.2.2.2.92 pre_rnd_fir.c

(./simnet/release/src/vehicle/libsrc/libmsg/pre_rnd_fir.c)

Includes:

```
"stdio.h"
"sim_types.h"
"sim_dfns.h"
"mass_stdh.h"
"dgi_stdh.h"
"sim_cig_if.h"
"libmsg.h"
```

2.1.2.2.2.92.1 cig_msg_prepend_ballistics_msg

This routine prepends the Round Fired message to the CIG message buffer. The parameters represent information that will be added to the buffer, and are defined as follows:

- | | |
|-------------------------|---|
| <i>type</i> | - round type. |
| <i>tracer</i> | - tracer effect. A value of 0 will cause the effect type in the trajectory table specified by round type to be displayed as the tracer; otherwise, a valid tracer effect must be specified. |
| <i>id</i> | - unique round identifier. |
| <i>gunpos</i> | - the gun position, in world coordinates, at the time of firing. |
| <i>gunvel</i> | - the gun elevation, in world coordinates, at the time of firing. |
| <i>sinelv</i> | - sine of the gun elevation angle, in world coordinates, at the time of firing. |
| <i>coselv</i> | - cosine of the gun elevation angle, in world coordinates, at the time of firing. |
| <i>sinazm</i> | - sine of the gun azimuth angle, in world coordinates, at the time of firing. |
| <i>cosazm</i> | - cosine of the gun azimuth angle, in world coordinates, at the time of firing. |
| <i>est_impact_time</i> | - estimated impact time. |
| <i>est_impact_range</i> | - estimated impact range. |

Parameters		
Parameter	Type	Where Typedef Declared
type	int	Standard
tracer	int	Standard
id	int	Standard
gunpos	VECTOR	sim_types.h
gunvel	VECTOR	sim_types.h
sinelv	REAL	sim_types.h
coselv	REAL	sim_types.h
sinazm	REAL	sim_types.h
cosazm	REAL	sim_types.h
est_impact_time	REAL	sim_types.h
est_impact_range	REAL	sim_types.h
Internal Variables		
Internal Variable	Type	Where Typedef Declared
mp	register pointer to MSG ROUND FIRED	sim_cig_if.h
Calls		
Function	Where Described	
prepend_msg_hdr	Section 2.1.2.2.2.83.1	

Table 2.1-266: cig_msg_prepend_ballistics_msg Information.

2.1.2.2.2.93 pre_rot2x1.c

(./simnet/release/src/vehicle/libsrc/libmsg/pre_rot2x1.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_std.h"
"dgi_std.h"
"sim_cig_if.h"
"libmsg.h"
```

2.1.2.2.2.93.1 cig_msg_prepend_rot2x1_matrix

This routine prepends the Rot2x1 Matrix message to the CIG message buffer. The parameters represent information that will be added to the buffer, and are defined as follows:

<i>node_index</i>	-	the value representing the node that will be updated by the matrix.
<i>cos_rot</i>	-	the cosine of the angle of rotation.
<i>sin_rot</i>	-	the sine of the angle of rotation.
<i>axis</i>	-	the axis of rotation; it will have one of the following values:
	0	heading
	1	pitch
	2	roll

Parameters		
Parameter	Type	Where Typedef Declared
node_index	WORD	mass_std.h
cos_rot	REAL 4	mass_std.h
sin_rot	REAL 4	mass_std.h
axis	BYTE	sim_types.h
Internal Variables		
Internal Variable	Type	Where Typedef Declared
mp	register pointer to MSG ROT2x1 MATRIX	sim_cig_if.h
Calls		
Function	Where Described	
prepend_msg_hdr	Section 2.1.2.2.2.83.1	

Table 2.1-267: cig_msg_prepend_rot2x1_matrix Information.

2.1.2.2.2.94 pre_rts4x3.c

(.simnet/release/src/vehicle/libsrc/libmsg/pre_rts4x3.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_std.h"
"dgi_std.h"
"sim_cig_if.h"
"libmsg.h"
```

2.1.2.2.2.94.1 cig_msg_prepend_rts4x3_matrix

This routine prepends the RTS4x3 Matrix message to the CIG message buffer. The parameters represent information that will be added to the buffer, and are defined as follows:

node_index - the node that will be updated by the 4x3 matrix.
rot_mtx - the transformation matrix that describes the location of the vehicle.
vec - X, Y, Z translation values that describe the location of the vehicle.

Parameters		
Parameter	Type	Where Typedef Declared
<i>node_index</i>	int	Standard
<i>rot_mtx</i>	T_MATRIX	sim_types.h
<i>vec</i>	VECTOR	sim_types.h
Internal Variables		
Internal Variable	Type	Where Typedef Declared
<i>mp</i>	register pointer to MSG_ROT4x3_MATRIX	sim_cig_if.h
Calls		
Function	Where Described	
prepend_msg_hdr	Section 2.1.2.2.2.83.1	

Table 2.1-268: **cig_msg_prepend_rts4x3_matrix** Information.

2.1.2.2.2.95 pre_scale.c

(/simnet/release/src/vehicle/libsrc/libmsg/pre_scale.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_stdc.h"
"dgi_stdg.h"
"sim_cig_if.h"
"libmsg.h"
```

2.1.2.2.2.95.1 cig_msg_prepend_scale

This routine prepends the Scale Message to the CIG message buffer. The parameters represent information that will be added to the buffer, and are defined as follows:

node_index - the node that will be updated by the following parameter.
scalep - three values that describe the new x-scale, y-scale, and z-scale, to be placed in the new HPRXYZS matrix.

Parameters		
Parameter	Type	Where Typedef Declared
<i>node_index</i>	WORD	mass_stdc.h
<i>scalep</i>	pointer to R4P3D	dgi_stdg.h
Internal Variables		
Internal Variable	Type	Where Typedef Declared
<i>mp</i>	register pointer to MSG SCALE	sim_cig_if.h
Calls		
Function	Where Described	
<i>prepend_msg_hdr</i>	Section 2.1.2.2.2.83.1	

Table 2.1-269: *cig_msg_prepend_scale* Information.

2.1.2.2.2.96 pre_show_eff.c

(/simnet/release/src/vehicle/libsrc/libmsg/pre_show_eff.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_std.h"
"dgi_std.h"
"sim_cig_if.h"
"libmsg.h"
```

2.1.2.2.2.96.1 cig_msg_prepend_show_effect

This routine prepends the Show Effect message to the CIG message buffer. The parameters represent information that will be added to the buffer, and are defined as follows:

id - an identifier for active effects.
type - the effect type.
tl - application-specific identification data.

Parameters		
Parameter	Type	Where Typedef Declared
<i>id</i>	int	Standard
<i>type</i>	int	Standard
<i>tl</i>	pointer to R4P3D	dgi_std.h
Internal Variables		
Internal Variable	Type	Where Typedef Declared
<i>mp</i>	register pointer to MSG SHOW EFFECT	sim_cig_if.h
Calls		
Function	Where Described	
prepend_msg_hdr	Section 2.1.2.2.2.83.1	

Table 2.1-270: cig_msg_prepend_show_effect Information.

2.1.2.2.2.97 pre_stat_rm.c

(./simnet/release/src/vehicle/libsrc/libmsg/pre_stat_rm.c)

Includes:

```
"stdio.h"
"sim_dfns.h"
"sim_macros.h"
"sim_types.h"
"mass_stdc.h"
"dgi_stdg.h"
"sim_cig_if.h"
"libveh.h"
"pro_sim.h"
"librva.h"
"libmsg.h"
"msg_loc.h"
"libmap.h"
```

2.1.2.2.2.97.1 cig_msg_prepend_staticveh_rem

This routine prepends the Static Vehicle Remove message to the CIG message buffer. *veh_list* is a pointer to the list of RVA entries, which contains information about the location of static vehicles, and *num_vehs* is the number of vehicles that will be removed.

Parameters		
Parameter	Type	Where Typedef Declared
veh_list[]	pointer to RVA_ENTRY	librva.h
num_vehs	int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
vap	pointer to VehicleAppearanceVariant	p_sim.h
k	register int	Standard
r_pkt	pointer to RVA_ENTRY	librva.h
mp	register pointer to MSG_STATICVEH_REM	sim_cig_if.h
from	register pointer to REAL	sim_types.h
to	register pointer to REAL 4	mass_stdc.h
Calls		
Function	Where Described	
prepend_msg_hdr	Section 2.1.2.2.2.83.1	
network_get_vehicle_force	Section 2.1.1.3.1.17.1	
map_net_to_cig	Section 2.6.11.5.8	

Table 2.1-271: cig_msg_prepend_staticveh_rem Information.

2.1.2.2.2.98 pre_stat_veh.c

(./simnet/release/src/vehicle/libsrc/libmsg/pre_stat_veh.c)

Includes:

```
"stdio.h"
"sim_dfns.h"
"sim_macros.h"
"sim_types.h"
"mass_stdc.h"
"dgi_stdg.h"
"sim_cig_if.h"
"libveh.h"
"pro_sim.h"
"librva.h"
"libmsg.h"
"msg_loc.h"
"libmap.h"
```

2.1.2.2.2.98.1 cig_msg_prepend_staticveh_state

This routine prepends the Static Vehicle State message to the CIG message buffer. *veh_list* is a pointer to the list of RVA entries, which contains information about static vehicles, and *num_vehs* is the number of static vehicles that are to be changed.

Parameters		
Parameter	Type	Where Typedef Declared
veh_list[]	pointer to RVA_ENTRY	librva.h
num_vehs	int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
vap	pointer to VehicleAppearanceVariant	p_sim.h
k	register int	Standard
r_pkt	pointer to RVA_ENTRY	librva.h
mp	register pointer to MSG_STATICVEH_STATE	sim_cig_if.h
from	register pointer to float	Standard
to	register pointer to REAL_4	mass_stdc.h
Calls		
Function	Where Described	
prepend_msg_hdr	Section 2.1.2.2.2.83.1	
network_get_vehicle_force	Section 2.1.1.3.1.17.1	
map_format_asid	Section 2.6.11.4.7	
map_net_to_cig	Section 2.6.11.5.8	

Table 2.1-272: cig_msg_prepend_staticveh_state Information.

2.1.2.2.2.99 pre_submode.c

(/simnet/release/src/vehicle/libsrc/libmsg/pre_submode.c)

This file is not implemented in the Masscomp or Butterfly.

2.1.2.2.2.100 pre_sys_err.c

(/simnet/release/src/vehicle/libsrc/libmsg/pre_sys_err.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_stdh.h"
"dgi_stdh.h"
"sim_cig_if.h"
"libmsg.h"
```

2.1.2.2.2.100.1 cig_msg_prepend_sys_error

This routine prepends the System Error message to the CIG message buffer. The parameters represent information that will be added to the buffer, and are defined as follows:

error_msg - error message type
cig_state - CIG state

Parameters		
Parameter	Type	Where Typedef Declared
<i>error_msg</i>	int	Standard
<i>cig_state</i>	int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
<i>mp</i>	register pointer to MSG_STATICVEH REM	sim_cig_if.h
Calls		
Function	Where Described	
<i>prepend_msg_hdr</i>	Section 2.1.2.2.2.83.1	

Table 2.1-273: *cig_msg_prepend_sys_error* Information.

2.1.2.2.2.101 pre_test_nam.c (./simnet/release/src/vehicle/libsrc/libmsg/pre_test_nam.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_std.h"
"dgi_std.h"
"sim_cig_if.h"
"libmsg.h"
```

2.1.2.2.2.101.1 cig_msg_prepend_test_name

This routine prepends the Test Name message to the CIG message buffer. *test_number* is the test identifier.

Parameters		
Parameter	Type	Where Typedef Declared
test_number	int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
mp	register pointer to MSG TEST_NAME	sim_cig_if.h
Calls		
Function	Where Described	
prepend_msg_hdr	Section 2.1.2.2.2.83.1	

Table 2.1-274: cig_msg_prepend_test_name Information.

2.1.2.2.2.102 pre_traj_chd.c (./simnet/release/src/vehicle/libsrc/libmsg/pre_traj_chd.c)

Includes:

```
"stdio.h"
"sim_types.h"
"sim_dfns.h"
"mass_stdh.h"
"dgi_stdg.h"
"sim_cig_if.h"
"libmsg.h"
```

2.1.2.2.2.102.1 cig_msg_prepend_traj_chord

This routine prepends the Trajectory Chord message to the CIG message buffer. The parameters represent information that will be added to the buffer, and are defined as follows:

- type* - the chord type; it generally represents an ammunition or missile type.
- id* - an identifier for the chord.
- tracer* - effect type. A value of 0 will prevent an effect from being displayed; otherwise, the value represents the effect type to be displayed as the tracer.
- begin* - the X-, Y-, and Z-coordinates that describe the starting position of the chord.
- end* - the X-, Y-, and Z-coordinates that describe the ending position of the chord.

Parameters		
Parameter	Type	Where Typedef Declared
<i>type</i>	int	Standard
<i>id</i>	int	Standard
<i>tracer</i>	BOOLEAN	sim_types.h
<i>begin</i>	VECTOR	sim_types.h
<i>end</i>	VECTOR	sim_types.h
Internal Variables		
Internal Variable	Type	Where Typedef Declared
<i>mp</i>	register pointer to MSG_TRAJ_CHORD	sim_cig_if.h
Calls		
Function	Where Described	
prepend_msg_hdr	Section 2.1.2.2.2.83.1	

Table 2.1-275: cig_msg_prepend_traj_chord Information.

2.1.2.2.2.103 pre_traj_ent.c (./simnet/release/src/vehicle/libsrc/libmsg/pre_traj_ent.c)

Includes:

```
"stdio.h"
"sim_types.h"
"sim_dfns.h"
"mass_std.h"
"dgi_stdg.h"
"sim_cig_if.h"
"libmsg.h"
```

2.1.2.2.2.103.1 cig_msg_prepend_traj_entry_xfer

This routine prepends the Trajectory Entry Transfer message to the CIG message buffer. The parameters represent information that will be added to the buffer, and are defined as follows:

bore_x - the horizontal displacement from the gun tip of this trajectory entry.
bore_z - the vertical displacement from the gun tip of this trajectory entry.

Parameters		
Parameter	Type	Where Typedef Declared
<i>bore_x</i>	REAL	sim_types.h
<i>bore_z</i>	REAL	sim_types.h
Internal Variables		
Internal Variable	Type	Where Typedef Declared
mp	register pointer to MSG TRAJ ENTRY XFER	sim_cig_if.h
Calls		
Function	Where Described	
prepend msg_hdr	Section 2.1.2.2.2.83.1	

Table 2.1-276: cig_msg_prepend_traj_entry_xfer Information.

2.1.2.2.2.104 pre_traj_tbl.c (./simnet/release/src/vehicle/libsrc/libmsg/pre_traj_tbl.c)

Includes:

```
"stdio.h"
"sim_types.h"
"sim_dfns.h"
"mass_stdh.h"
"dgi_stdg.h"
"sim_cig_if.h"
"libmsg.h"
```

2.1.2.2.2.104.1 cig_msg_prepend_traj_table_xfer

This routine prepends the Trajectory Table Transfer message to the CIG message buffer. The parameters represent information that will be added to the buffer, and are defined as follows:

ammo_type - the ammunition type of the tracer effect to be displayed.
traj_index - the trajectory table numbers.
count - size of the trajectory table.

Parameters		
Parameter	Type	Where Typedef Declared
ammo_type	int	Standard
traj_index	int	Standard
count	int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
mp	register pointer to MSG TRAJ TABLE XFER	sim_cig_if.h
Calls		
Function	Where Described	
prepend_msg_hdr	Section 2.1.2.2.2.83.1	

Table 2.1-277: cig_msg_prepend_traj_table_xfer Information.

2.1.2.2.2.105 pre_trans.c (./simnet/release/src/vehicle/libsrc/libmsg/pre_trans.c)

Includes:

```
"stdc.h"
"sim_types.h"
"sim_defs.h"
"msg_std.h"
"dgi_std.h"
"sim_cig_if.h"
"libmsg.h"
```

2.1.2.2.2.105.1 cig_msg_prepend_update_translation

This routine prepends the Translation Update message to the CIG message buffer. The parameters present information that will be added to the buffer, and are defined as follows:

- node_index* - the node that will be updated by the following parameter.
- transp* - pointer to xyz translation

Parameters		
Parameter	Type	Where Typedef Declared
node_index	HWND	mass_std.h
transp	pointer to R4P3D	dgi_std.h
Internal Variables		
Internal Variable	Type	Where Typedef Declared
mp	register pointer to MSG_TRANSLATION	sim_cig_if.h
Calls		
Function	Where Described	
prepend_msg_hdr	Section 2.1.2.2.2.83.1	

Table 2.1-278: cig_msg_prepend_update_translation Information.

2.1.2.2.2.106 pre_vflags.c (./simnet/release/src/vehicle/libsrc/libmsg/pre_vflags.c)

Includes:

```
"stdio.h"
"sim_types.h"
"sim_dfns.h"
"mass_std.h"
"dgi_std.h"
"sim_cig_if.h"
"libvflags.h"
"libmsg.h"
"msg_loc.h"
```

2.1.2.2.2.106.1 cig_msg_prepend_view_flags

This routine prepends the View Flags message to the CIG message buffer. The parameters represent information that will be added to the buffer, and are defined as follows:

- view_flags* - values that turn on or off individual processing paths in the CIG system.
- branch_values* - an array of values that control the branching of the branch nodes. Configuration branch nodes use values from this array to determine which branch to take. These values are compared against the branch mask specified in the Create Configuration Node message.

Parameters		
Parameter	Type	Where Typedef Declared
<i>view_flags</i>	WORD	mass_std.h
<i>branch_values[]</i>	WORD	mass_std.h
Internal Variables		
Internal Variable	Type	Where Typedef Declared
<i>mp</i>	register pointer to MSG_VIEW_FLAGS	sim_cig_if.h
Calls		
Function	Where Described	
<i>prepend_msg_hdr</i>	Section 2.1.2.2.2.83.1	

Table 2.1-279: cig_msg_prepend_view_flags Information.

2.1.2.2.2.107 pre_vmag.c (./simnet/release/src/vehicle/libsrc/libmsg/pre_vmag.c)

Includes:

```
"stdio.h"
"sim_types.h"
"sim_dfns.h"
"mass_stdc.h"
"dgi_stdg.h"
"sim_cig_if.h"
"libmsg.h"
```

2.1.2.2.2.107.1 cig_msg_prepend_view_magnification

This routine prepends the View Magnification message to the CIG message buffer. The parameters represent information that will be added to the buffer, and are defined as follows:

node_index - the node that will be updated by the following parameters.
lod_magnifier - the level-of-detail magnifier.
i - the horizontal field of view, in degrees.
j - the vertical field of view, in degrees.

Parameters		
Parameter	Type	Where Typedef Declared
<i>node_index</i>	WORD	mass_stdc.h
<i>lod_magnifier</i>	REAL_4	mass_stdc.h
<i>i</i>	REAL_4	mass_stdc.h
<i>j</i>	REAL_4	mass_stdc.h
Internal Variables		
Internal Variable	Type	Where Typedef Declared
<i>mp</i>	register pointer to MSG VIEW MAGNIFICATION	sim_cig_if.h
Calls		
Function	Where Described	
prepend msg_hdr	Section 2.1.2.2.2.83.1	

Table 2.1-280: cig_msg_prepend_view_magnification Information.

2.1.2.2.2.108 pre_vmode.c

(./simnet/release/src/vehicle/libsrc/libmsg/pre_vmode.c)

Includes:

"stdio.h"
 "sim_types.h"
 "sim_dfns.h"
 "mass_stdc.h"
 "dgi_stdg.h"
 "sim_cig_if.h"
 "libmsg.h"
 "msg_loc.h"

2.1.2.2.2.108.1 cig_msg_prepend_view_mode

This routine prepends the View Mode message to the CIG message buffer. The parameter *view_mode* represents information that will be added to the buffer.

Parameters		
Parameter	Type	Where Typedef Declared
view_mode	WORD	mass_stdc.h
Internal Variables		
Internal Variable	Type	Where Typedef Declared
mp	register pointer to MSG_VIEW_MODE	sim_cig_if.h
Calls		
Function	Where Described	
prepend_msg_hdr	Section 2.1.2.2.2.83.1	

Table 2.1-281: cig_msg_prepend_view_mode Information.

2.1.2.2.2.109 pre_vport.c (./simnet/release/src/vehicle/libsrc/libmsg/pre_vport.c)

Includes:

```
"stdio.h"
"sim_types.h"
"sim_dfns.h"
"mass_stdh.h"
"dgi_stdg.h"
"sim_cig_if.h"
"libmsg.h"
```

2.1.2.2.2.109.1 cig_msg_prepend_viewport_state

This routine prepends the Viewport State message to the CIG buffer. The parameters represent information that will be added to the buffer, and are defined as follows:

<i>node_index</i>	- defines the node to which the viewport is attached.
<i>viewport_id</i>	- the id value assigned to the viewport.
<i>database_id</i>	- the database id.
<i>res</i>	- the screen resolution.
<i>viewing_range</i>	- the distance that can be viewed from the viewport.
<i>near_plane</i>	- a value greater than or equal to 1 changes the near plane of the viewport.
<i>lod_magnifier</i>	- the level-of-detail multiplier. This value adjusts the level of detail ranges.
<i>aspect_ratio</i>	- the aspect ratio.
<i>i</i>	- the horizontal field of view, in degrees.
<i>j</i>	- the vertical field of view, in degrees.

Parameters		
Parameter	Type	Where Typedef Declared
<i>node_index</i>	HWORD	mass_stdh.h
<i>viewport_id</i>	BYTE	sim_types.h
<i>database_id</i>	BYTE	sim_types.h
<i>res</i>	pointer to RESOLUTION	sim_cig_if.h
<i>viewing_range</i>	REAL 4	mass_stdh.h
<i>near_plane</i>	REAL 4	mass_stdh.h
<i>lod_magnifier</i>	REAL 4	mass_stdh.h
<i>aspect_ratio</i>	REAL 4	mass_stdh.h
<i>i</i>	REAL 4	mass_stdh.h
<i>j</i>	REAL 4	mass_stdh.h
Internal Variables		
Internal Variable	Type	Where Typedef Declared
<i>mp</i>	register pointer to MSG_VIEWPORT_STATE	sim_cig_if.h
Calls		
Function	Where Described	
prepend_msg_hdr	Section 2.1.2.2.2.83.1	

Table 2.1-282: cig_msg_prepend_viewport_state Information.

2.1.2.2.2.110 **pre_vupdate.c** (./simnet/release/src/vehicle/libsrc/libmsg/pre_vupdate.c)

This file is not implemented in the Masscomp or Butterfly.

2.1.2.2.2.111 **printbuffer.c** (./simnet/release/src/vehicle/libsrc/libmsg/printbuffer.c)

This file contains a routine which prints the contents of each message buffer.

Includes:

```
"sim_types.h"
"mass_stdh.h"
"dgi_stdg.h"
"sim_cig_if.h"
"gbuffer.h"
```

2.1.2.2.2.111.1 **print_buffer**

This routine prints the name and contents of each message in the message buffer. It is used as a debugging tool. *mbp* is a pointer to the message buffer.

Parameters		
Parameter	Type	Where Typedef Declared
mbp	pointer to MSGS_BLK	sim_cig_if.h
Internal Variables		
Internal Variable	Type	Where Typedef Declared
bytes_done	int	Standard
hp	pointer to MSG_HDR	sim_cig_if.h
Calls		
Function	Where Described	
DOT	gbuffer.h (macro definition)	
print_msg_end	Section 2.1.2.2.2.44.1	
print_msg_cig_ctl	Section 2.1.2.2.2.41.1	
print_msg_other_veh_state	Section 2.1.2.2.2.53.1	
print_msg_laser_return	Section 2.1.2.2.2.50.1	
print_msg_traj_chord	Section 2.1.2.2.2.62.1	
print_msg_show_effect	Section 2.1.2.2.2.56.1	
print_msg_local_terrain	Section 2.1.2.2.2.51.1	
print_msg_test_name	Section 2.1.2.2.2.61.1	
print_msg_file_descr	Section 2.1.2.2.2.45.1	
print_msg_file_xfer	Section 2.1.2.2.2.47.1	
print_msg_file_status	Section 2.1.2.2.2.46.1	
print_msg_sys_error	Section 2.1.2.2.2.60.1	
print_msg_staticveh_state	Section 2.1.2.2.2.58.1	
print_msg_staticveh_rem	Section 2.1.2.2.2.57.1	
print_msg_round_fired	Section 2.1.2.2.2.55.1	
print_msg_agl	Section 2.1.2.2.2.40.1	

Table 2.1-283: **print_buffer** Information.

2.1.2.2.2.112 **set_assym.c** (./simnet/release/src/vehicle/libsrc/libmsg/set_assym.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_stdh.h"
"dgi_stdg.h"
"sim_cig_if.h"
"msg_loc.h"
```

2.1.2.2.2.112.1 **set_assymmetric_on**

This routine sets *using_assymmetric* to TRUE.

2.1.2.2.2.113 **set_buf_num.c** (./simnet/release/src/vehicle/libsrc/libmsg/set_buf_num.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_stdh.h"
"dgi_stdg.h"
"sim_cig_if.h"
"msg_loc.h"
```

2.1.2.2.2.113.1 **set_buffer_num**

This routine sets the buffer number to *num*.

Parameters		
Parameter	Type	Where Typedef Declared
num	int	Standard

Table 2.1-284: **set_buffer_num** Information.

2.1.2.2.2.114 set_cig_mask.c (./simnet/release/src/vehicle/libsrc/libmsg/set_cig_mask.c)

Includes:

```
"stdio.h"  
"sim_types.h"  
"mass_std.h"  
"dgi_std.h"  
"sim_cig_if.h"  
"msg_loc.h"
```

2.1.2.2.2.114.1 set_cig_mask

This routine sets bit wise ORs *num* to the current CIG mask.

Parameters		
Parameter	Type	Where Typedef Declared
num	int	Standard

Table 2.1-285: set_cig_mask Information.

2.1.2.2.2.115 set_veh_spec.c (./simnet/release/src/vehicle/libsrc/libmsg/set_veh_spec.c)

Includes:

```
"sim_types.h"  
"mass_std.h"  
"dgi_std.h"  
"sim_cig_if.h"  
"gbuffer.h"  
"cig_buffer.h"  
"libmsg.h"  
"msg_loc.h"
```

2.1.2.2.2.115.1 cig_set_veh_spec_ptr

This routine sets pointers to the location of CIG specific information in the message buffer. This routine is only called when two CIGs are being used. It is not implemented in Version 6.6 of this code.

2.1.2.2.2.116 use_debug.c (./simnet/release/src/vehicle/libsrc/libmsg/use_debug.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_std.c.h"
"dgi_stdg.h"
"sim_cig_if.h"
"msg_loc.h"
```

2.1.2.2.2.116.1 use_static_debug

This routine sets the flag *on* such that debugging of static vehicles is enabled. It is a debugging tool.

Parameters		
Parameter	Type	Where Typedef Declared
<i>on</i>	int	Standard

Table 2.1-286: use_static_debug Information.

2.1.2.2.2.117 libmsg.h (./simnet/release/src/vehicle/libsrc/libmsg/libmsg.h)

The following routines are declared external:

```
prepend_msg_hdr()
deallocate_prepend_buffer_space()
append_msg_hdr()
deallocate_appended_buffer_space()
push_msg_cig_ctl()
cig_msg_prepend_ballistics_msg()
cig_msg_prepend_traj_chord()
cig_msg_prepend_eo()
cig_msg_prepend_show_effect()
push_msg_test_name()
cig_msg_append_end()
push_msg_laser_return()
push_msg_hit()
push_msg_sys_err()
push_msg_rtn_lt()
push_msg_file_descr()
push_msg_file_xfer()
push_msg_file_status()
get_static_debug()
use_static_debug()
cig_msg_append_staticveh_state()
cig_msg_append_asid_staticveh_state()
cig_append_staticveh_rem()
cig_adjust_for_changed_staticveh()
cig_msg_prepend_genveh_state()
```

```
cig_msg_prepend_request_laser_range()
cig_msg_prepend_pass_on()
cig_msg_prepend_pass_back()
cig_msg_prepend_zoom()
cig_msg_prepend_viewport_state()
cig_msg_prepend_view_mode()
cig_msg_prepend_view_flags()
cig_msg_append_view_flags()
cig_msg_prepend_view_magnification()
cig_msg_prepend_rot2x1_matrix()
cig_msg_prepend_rts4x3_matrix()
cig_msg_prepend_hprxyzs_matrix()
cig_msg_prepend_update_translation()
cig_msg_prepend_scale()
cig_msg_prepend_3rotations()
cig_msg_prepend_1rotation()
cig_msg_prepend_gen_configtree()
cig_msg_prepend_cig_config()
cig_msg_append_traj_table_xfer()
cig_msg_append_traj_entry_xfer()
cig_msg_prepend_agl_setup()
print_buffer()
check_buffer()
buffer_reset()
cig_flush_buffer()
cig_prepend_overall_header()
flush_buffer()
clear_n_mapped()
get_n_mapped()
cig_msg_adjust_otherveh_state()
append_other_in_send_buffer()
add_veh_to_cig_msg()
delete_veh_from_cig_msg()
print_msg_agl()
print_msg_cig_ctl()
print_msg_end()
print_msg_eo()
print_msg_file_descr()
print_msg_file_status()
print_msg_file_xfer()
print_msg_hit()
print_msg_hit_return()
print_msg_miss()
print_msg_laser_return()
print_msg_local_terrain()
print_msg_m2veh_state()
print_msg_myveh_state()
print_msg_otherveh_state()
print_msg_round_fired()
print_msg_process_round()
print_msg_rtn_lt()
print_msg_show_effect()
print_msg_staticveh_rem()
print_msg_sys_error()
```

```
print_msg_test_name()
print_msg_traj_chord()
get_front_of_send_buffer()
get_other_start_in_send_buffer()
get_back_of_send_buffer()
cig_read_configfile()
cig_msg_prepend_gun_overlay()
cig_msg_prepend_overlay_setup()
set_number_of_cigs()
get_sbuffer()
get_init_ptrs()
buffer_setup()
cig_set_vch_spec_ptrs()
set_buffer_num()
multi_cig_msg_prepend_rts4x3_matrix()
multi_cig_msg_prepend_view_flags()
multi_push_msg_cig_ctl()
multi_push_msg_file_descr()
multi_cig_msg_append_end()
multi_cig_msg_prepend_pass_on()
multi_cig_msg_append_traj_table_xfer()
multi_cig_msg_append_traj_entry_xfer()
set_cig_mask()
get_cig_mask()
store_round_fired()
store_traj_chord()
init_ballistics_buffer()
copy_ballistics_buffer()
```

The following constants are defined:

```
CIG_NONE
CIG_1
CIG_2
CIG_ALL
```

The structure S_BUFFER is defined.

2.1.2.2.3 libproc

(./simnet/release/src/vehicle/libsrc/libproc [libproc])

Libproc contains the routines that are used to process messages received from the CIG.

2.1.2.2.3.1 alt_abv_gnd.c

(./simnet/release/src/vehicle/libsrc/libproc/alt_abv_gnd.c)

Includes:

"stdio.h"
"sim_dfns.h"
"sim_macros.h"
"sim_types.h"
"mass_stdh.h"
"dgi_stdg.h"
"sim_cig_if.h"
"proc_loc.h"

2.1.2.2.3.1.1 cig_altitude_above_gnd

This routine returns the altitude above ground from the MSG_AGL CIG message, which is returned by the CIG every frame. The altitude above ground is the height above ground of the vehicle centroid. This data is stored in libproc for access by any of the other software modules.

Return Values		
Return Value	Type	Meaning
altitude	REAL 4	the altitude above ground

Table 2.1-287: cig_altitude_above_gnd Information.

2.1.2.2.3.2 `get_f_status.c` (./simnet/release/src/vehicle/libsrc/libproc/get_f_status.c)

Includes:

```
"stdio.h"  
"sim_types.h"  
"mass_std.h"  
"dgi_stdg.h"  
"sim_cig_if.h"  
"proc_loc.h"
```

2.1.2.2.3.2.1 `cig_get_file_status_data`

This routine obtains data about the file status by returning a pointer to the MSG_FILE_STATUS CIG message. The file status message is processed in the routine `process_msg_file_status()` in "proc_f_stat.c".

Internal Variables		
Variable	Type	Where Typedef Declared
ret	pointer to MSG FILE STATUS	sim_cig_if.h
Return Values		
Return Value	Type	Meaning
ret	pointer to MSG FILE STATUS	data on the file status

Table 2.1-288: `cig_get_file_status_data` Information.

2.1.2.2.3.3 get_file_dat.c

(./simnet/release/src/vehicle/libsrc/libproc/get_file_dat.c)

Includes:

"stdio.h"
 "sim_types.h"
 "mass_stdh.h"
 "dgi_stdh.h"
 "sim_cig_if.h"
 "proc_loc.h"

2.1.2.2.3.3.1 cig_get_file_xfer_data

This routine obtains data on the file transfer by returning a pointer to the MSG_FILE_XFER CIG message. The file transfer message is processed in the routine process_msg_file_xfer() in "proc_f_stat.c".

Internal Variables		
Variable	Type	Where Typedef Declared
ret	pointer to MSG_FILE_XFER	sim_cig_if.h
Return Values		
Return Value	Type	Meaning
ret	pointer to MSG_FILE_XFER	data on the file transfer

Table 2.1-289: cig_get_file_xfer_data Information.

2.1.2.2.3.4 get_laser.c

(./simnet/release/src/vehicle/libsrc/libproc/get_laser.c)

Includes:

"stdio.h"
 "sim_types.h"
 "mass_stdh.h"
 "dgi_stdh.h"
 "sim_cig_if.h"
 "proc_loc.h"

2.1.2.2.3.4.1 cig_laser_range

This routine returns the laser range value from the MSG_LASER_RETURN CIG message. This data is stored in libproc for access by any of the other software modules.

Return Values		
Return Value	Type	Meaning
laser range value	REAL 4	mass_stdh.h

Table 2.1-290: cig_laser_range Information.

2.1.2.2.3.5 get_laser2.c

(/simnet/release/src/vehicle/libsrc/libproc/get_laser2.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_stdc.h"
"dgi_stdg.h"
"sim_cig_if.h"
"proc_loc.h"
```

2.1.2.2.3.5.1 cig_laser_range2

This routine returns the laser range value from the second CIG in a two-CIG configuration. This data is stored in libproc for access by any of the other software modules.

Return Values		
Return Value	Type	Meaning
laser range value2	REAL 4	mass_stdc.h

Table 2.1-291: cig_laser_range2 Information.

2.1.2.2.3.6 init_agl_rtn.c

(/simnet/release/src/vehicle/libsrc/libproc/init_agl_rtn.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_stdc.h"
"dgi_stdg.h"
"sim_cig_if.h"
"proc_loc.h"
```

2.1.2.2.3.6.1 cig_init_msg_agl_routine

This routine initializes the above ground level routine. The routine takes **rtn()** as a parameter where **rtn** is the name of a function that returns an integer.

Parameters		
Parameter	Type	Where Typedef Declared
rtn	PFI	sim_types.h

Table 2.1-292: cig_init_msg_agl_routine Information.

2.1.2.2.3.7 **proc_agl.c** (./simnet/release/src/vehicle/libsrc/libproc/proc_agl.c)

Includes:

```
"stdio.h"  
"sim_dfns.h"  
"sim_macros.h"  
"sim_types.h"  
"mass_std.h"  
"dgi_stdg.h"  
"sim_cig_if.h"  
"proc_loc.h"
```

2.1.2.2.3.7.1 **process_msg_agl**

This routine processes the MSG_AGL CIG message. It calls the routine that was set in **cig_init_msg_agl_routine()**, passing it the variable *altitude*, which is the current altitude above ground.

Parameters		
Parameter	Type	Where Typedef Declared
mp	pointer to MSG_AGL	sim_cig_if.h

Table 2.1-293: **process_msg_agl** Information.

2.1.2.2.3.8 **proc_buf.c** (./simnet/release/src/vehicle/libsrc/libproc/proc_buf.c)

Includes:

```
"sim_types.h"  
"dgi_stdh.h"  
"dgi_stdg.h"  
"sim_cig_if.h"  
"if_ctas.h"  
"gbuffer.h"  
"proc_loc.h"  
"libmsg.h"  
"pro_sim.h"
```

External declarations: debug

2.1.2.2.3.8.1 process_buffer

This routine steps through each of the messages in the CIG buffer and determines the appropriate processing for the type of message.

Parameters		
Parameter	Type	Where Typedef Declared
mbp	pointer to MSGS_BLK	sim cig if.h
buffer_num	int	Standard
Internal Variables		
Variable	Type	Where Typedef Declared
msg_count	int	Standard
app_pkt	external pointer to SimulationPDU	p_sim.h
bytes_done	int	Standard
blk_size	int	Standard
Calls		
Function	Where Described	
BYTE COUNT	gbuffer.h	
DOT	gbuffer.h	
process_msg_laser_return	Section 2.1.2.2.3.16.1	
process_msg_sys_error	Section 2.1.2.2.3.22.1	
is_air_vehicle	Section 2.6.10.1.1	
process_msg_local_terrain	Section 2.1.2.2.3.15.1	
process_msg_lt_piece	Section 2.1.2.2.3.19.1	
get_ballistics_debug	Section 2.1.2.2.2.44.1	
print_msg_miss	Section 2.1.2.2.2.52.1	
process_msg_miss	Section 2.1.2.2.3.20.1	
print_msg_hit_return	Section 2.1.2.2.2.49.1	
process_msg_hit_return	Section 2.1.2.2.3.14.1	
process_msg_agl	Section 2.1.2.2.3.7.1	
process_msg_file_descr	Section 2.1.2.2.3.13.1	
process_msg_file_xfer	Section 2.1.2.2.3.12.1	
process_msg_file_status	Section 2.1.2.2.3.11.1	
process_msg_pass_back	Section 2.1.2.2.3.21	

Table 2.1-294: process_buffer Information.

2.1.2.2.3.9 proc_ct_ram.c

(./simnet/release/src/vehicle/libsrc/libproc/proc_ct_ram.c)

The code in this file is applicable only to the GT machine, which is not included with this system.

2.1.2.2.3.10 **proc_end.c** (./simnet/release/src/vehicle/libsrc/libproc/proc_end.c)

This routine processes the end message from the CIG buffer. This message serves as a signal to stop processing because the messages have ended.

2.1.2.2.3.11 **proc_f_stat.c** (./simnet/release/src/vehicle/libsrc/libproc/proc_f_stat.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_std.h"
"dgi_stdg.h"
"sim_cig_if.h"
"proc_loc.h"
```

2.1.2.2.3.11.1 **process_msg_file_status**

This routine processes the MSG_FILE_STATUS CIG message. This message is used to acknowledge that a block has been received or to request re-transmission of that block. The routine saves a pointer to the file status data. This pointer, *mp*, is only valid for the current frame and will be rewritten during the next tick.

Parameters		
Parameter	Type	Where Typedef Declared
mp	pointer to MSG_FILE_STATUS	sim_cig_if.h
length	int	Standard

Table 2.1-295: process_msg_file_status Information.

2.1.2.2.3.12 **proc_f_xfer.c** (/simnet/release/src/vehicle/libsrc/libproc/proc_f_xfer.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_stdh.h"
"dgi_stdg.h"
"sim_cig_if.h"
"proc_loc.h"
```

2.1.2.2.3.12.1 **process_msg_file_xfer**

This routine processes the MSG_FILE_XFER CIG message. This message is used to transfer files and databases. The routine saves a pointer to the file transfer data. This pointer is only valid for the current frame and will be rewritten during the next tick.

Parameters		
Parameter	Type	Where Typedef Declared
mp	pointer to MSG_FILE_XFER	sim_cig_if.h
length	int	Standard
Internal Variables		
Variable	Type	Where Typedef Declared
blk size	int	Standard
Return Values		
Return Value	Type	Meaning
blk size+1	int	the bulk size of the file transfer

Table 2.1-296: **process_msg_file_xfer** Information.

2.1.2.2.3.13 **proc_fdescr.c** (./simnet/release/src/vehicle/libsrc/libproc/proc_fdescr.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_std.c.h"
"dgi_stdg.h"
"sim_cig_if.h"
```

2.1.2.2.3.13.1 **process_msg_file_descr**

This routine processes the MSG_FILE_DESCR CIG message. This message contains information about files or databases that will be transferred. This message is also used to provide a directory listing to the simulation host, to delete files from the CIG, or to specify the simulation database.

Parameters		
Parameter	Type	Where Typedef Declared
mp	pointer to MSG_FILE_DESCR	sim_cig_if.h
length	int	Standard
Calls		
Function	Where Described	
print_msg_file_descr	Section 2.1.2.2.2.45.1	

Table 2.1-297: **process_msg_file_descr** Information.

2.1.2.2.3.14 **proc_hit.c** (./simnet/release/src/vehicle/libsrc/libproc/proc_hit.c)

Includes:

```
"stdio.h"
"sim_dfns.h"
"sim_macros.h"
"sim_types.h"
"mun_type.h"
"mass_std.c.h"
"dgi_stdg.h"
"sim_cig_if.h"
"libmap.h"
"libimps.h"
"libmatrix.h"
"libkin.h"
"libhull.h"
"libmissile.h"
"libnetwork.h"
"librva.h"
"proc_loc.h"
```

Declarations:

```
proc_hit_debug = FALSE
```

2.1.2.2.3.14.1 process_msg_hit_return

This routine processes the MSG_HIT_RETURN CIG hit message. This message returns ballistics intersection information. The message contains information on the type of projectile that was fired and the type of object that was struck.

For missile impacts, the routine determines whether the impact was with the ground, with a vehicle hull or turret, or with an unknown object. Ground impacts call the routine `missile_util_comm_intersected_poly()` for processing. Vehicle impacts call the routine `missile_util_comm_intersected_model()` for processing. Unknown impacts return an error message.

For impacts other than those of missiles, the routine determines the range and the type of impact. Ground impacts call the routine `network_send_ground_impact()`. Vehicle impacts call the routine `network_send_vehicle_impact()`. Unknown impacts return an error message.

Parameters		
Parameter	Type	Where Typedef Declared
mp	register pointer to MSG_HIT_RETURN	sim_cig_if.h
length	int	Standard
Internal Variables		
Variable	Type	Where Typedef Declared
d_impact_pt	VECTOR	sim_types.h
r_2	REAL	sim_types.h
ammo_index	int	Standard
impact_pt	pointer to float	Standard
chord_start	pointer to float	Standard
chord_end	pointer to float	Standard
Calls		
Function	Where Described	
f2d_vec_copy	Section 2.6.2.10.1	
map_is_missile	Section 2.6.11.2.15	
missile_util_comm_intersected_poly	Section 2.5.3.25.4	
missile_util_comm_intersected_model	Section 2.5.3.25.5	
rva_get_veh_id	Section 2.5.12.10.1	
kinematics_range_squared	Section 2.5.8.10.1	
network_send_ground_impact	Section 2.1.1.3.1.25.1	
impacts_queue_effect	Section 2.5.15.1.3	
network_send_vehicle_impact	Section 2.1.1.3.1.70.1	

Table 2.1-298: process_msg_hit_return Information.

2.1.2.2.3.14.2 process_msg_hit

This routine prints the statement "got a msg_hit".

Parameters		
Parameter	Type	Where Typedef Declared
mp	register pointer to MSG_HIT	sim_cig_if.h
length	int	Standard

Table 2.1-299: process_msg_hit Information.

2.1.2.2.3.15 proc_l_terr.c

(./simnet/release/src/vehicle/libsrc/libproc/proc_l_terr.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_std.h"
"dgi_stdg.h"
"sim_cig_if.h"
"proc_loc.h"
"libveh.h"
"libkin.h"
"libhull.h"
```

2.1.2.2.3.15.1 process_msg_local_terrain

This routine processes the MSG_LOCAL_TERRAIN message, which is sent in pieces by the CIG over a series of frames (the default is 32 frames). This message contains information that the simulation host uses to determine the position and orientation of the simulated vehicle and to calculate the dynamics of that vehicle. The routine first preprocesses the bvols and polys, and then calls `terrain_preproc_terrain()`, located in `libterrain`.

Parameters		
Parameter	Type	Where Typedef Declared
mp	register pointer to MSG_LOCAL_TERRAIN	sim_cig_if.h
length	int	Standard
Internal Variables		
Variable	Type	Where Typedef Declared
poly	pointer to LT POLY ENTRY	sim_cig_if.h
bvol	pointer to LT BVOL ENTRY	sim_cig_if.h
Calls		
Function	Where Described	
terrain_preproc_terrain	Section 2.5.11.6.1	
kinematics_get_o_to_h	Section 2.5.8.2.4	

Table 2.1-300: process_msg_local_terrain Information.

2.1.2.2.3.16 proc_laser.c

(./simnet/release/src/vehicle/libsrc/libproc/proc_laser.c)

Includes:

"stdio.h"
 "sim_types.h"
 "mass_std.h"
 "dgi_stdg.h"
 "sim_cig_if.h"
 "proc_loc.h"

2.1.2.2.3.16.1 process_msg_laser_return

This routine processes the MSG_LASER_RETURN message, which is sent from the CIG every frame. This message contains laser range information for the specified pixel within the screen. The variable *laser_range_value* is set from the information passed in the message. This routine has the capability of handling two CIGs, although this system only includes one CIG.

Parameters		
Parameter	Type	Where Typedef Declared
mp	pointer to MSG_LASER_RETURN	sim_cig_if.h
buffer_num	int	Standard

Table 2.1-301: process_msg_laser_return Information.

2.1.2.2.3.17 proc_loc.c

(./simnet/release/src/vehicle/libsrc/libproc/proc_loc.c)

Includes:

"stdio.h"
 "sim_types.h"
 "mass_std.h"
 "dgi_stdg.h"
 "sim_cig.h"
 "proc_loc.h"

Declarations:

altitude
 msg_agl_rtn
 laser_range_value
 laser_range_value2
 file_xfer_data
 file_status_data
 lt_buffer[20]
 lt_chunk_size

This file contains declarations of variables.

2.1.2.2.3.18 **proc_loc.h** (./simnet/release/src/vehicle/libsrc/libproc/proc_loc.h)

This file contains declarations of variables.

2.1.2.2.3.19 **proc_lt_pi.c** (./simnet/release/src/vehicle/libsrc/libproc/proc_lt_pi.c)

Includes:

```
"stdio.h"
"sim_types.h"
"mass_std.h"
"dg_std.h"
"sim_cig_if.h"
"proc_loc.h"
"libveh.h"
"libkin.h"
"libhull.h"
```

2.1.2.2.3.19.1 **process_msg_lt_piece**

This routine processes the MSG_LT_PIECE message, which is sent by the CIG once per frame. The local terrain message is divided into pieces, which are sent as MSG_LT_PIECEs at a rate of one per frame. Once all of the pieces have been received, **process_msg_local_terrain()** is called to process the entire message.

Parameters		
Parameter	Type	Where Typedef Declared
mp	register pointer to MSG_LT_PIECE	sim_cig_if.h
length	int	Standard
Internal Variables		
Variable	Type	Where Typedef Declared
ctr	int	Standard
ltp	pointer to MSG_LOCAL_TERRAIN	sim_cig_if.h
chunkp	pointer to char	Standard
Calls		
Function	Where Described	
process_msg_local_terrain	Section 2.1.2.2.3.15.1	

Table 2.1-302: **process_msg_lt_piece** Information.

2.1.2.2.3.20 **proc_miss.c** (./simnet/release/src/vehicle/libsrc/libproc/proc_miss.c)

Includes:

```
"stdio.h"
"sim_dfns.h"
"sim_macros.h"
"sim_types.h"
"mass_stdh.h"
"dgi_stdg.h"
"sim_cig_if.h"
"proc_loc.h"
"libmap.h"
"libnetwork.h"
```

2.1.2.2.3.20.1 **process_msg_miss**

This routine processes the MSG_MISS CIG message. A miss occurs when a fired round does not hit anything (i.e., it passes out of the range of the terrain database). If the miss was by a missile, libmissile processes the message; otherwise, the routine **network_send_non_impact()** is called to process the message.

Parameters		
Parameter	Type	Where Typedef Declared
mp	pointer to MSG_MISS	sim_cig_if.h
length	int	Standard
Calls		
Function	Where Described	
map_is_missile	Section 2.6.11.2.15	
network_send_non_impact	Section 2.1.1.3.1.33.1	

Table 2.1-303: **process_msg_miss** Information.

2.1.2.2.3.21 **proc_pback.c** (./simnet/release/src/vehicle/libsrc/libproc/proc_pback.c)

The message processed by this file is not generated by the CIG used in this system.

2.1.2.2.3.22 proc_sys_err.c

(./simnet/release/src/vehicle/libsrc/libproc/proc_sys_err.c)

Includes:

"stdio.h"
 "sim_types.h"
 "mass_std.h"
 "dgi_std.h"
 "sim_cig_if.h"
 "libcig.h"

2.1.2.2.3.22.1 process_msg_sys_error

This routine processes the MSG_SYS_ERROR message from the CIG. The system error message is sent to the simulation every frame, informing it of whether or not an error exists. This routine determines how to process the various types of errors.

Parameters		
Parameter	Type	Where Typedef Declared
mp	pointer to MSG_SYS_ERROR	sim_cig_if.h
length	int	Standard
Internal Variables		
Variable	Type	Where Typedef Declared
local int	int	Standard
debug	extern int	Standard
Calls		
Function	Where Described	
cig too many vehicles	Section 2.5.12.28.1	
print msg sys error	Section 2.1.2.2.2.60.1	

Table 2.1-304: process_msg_sys_error Information.

2.1.2.2.3.23 set_chunk.c**(./simnet/release/src/vehicle/libsrc/libproc/set_chunk.c)**

Includes:

"stdio.h"
"sim_types.h"
"mass_std.c.h"
"dgi_stdg.h"
"sim_cig.if.h"
"proc_loc.h"

2.1.2.2.3.23.1 set_chunk_size

This routine sets the size of the piece (or chunk) of terrain in **process_msg_lt_piece()**, which is in "proc_lt_pi.c". It may be desirable to adjust the chunk size if the CIG buffer size has been changed from the default.

Parameters		
Parameter	Type	Where Typedef Declared
size	int	Standard
Return Values		
Return Value	Type	Meaning
size	int	Standard

Table 2.1-305: set_chunk_size Information.

2.1.2.2.4 libvflags

(./simnet/release/src/vehicle/libsrc/libvflags [libvflags])

Information for the `view_flags` and `view_modes` messages is contained in `libvflags`. These messages control which screens are turned on and which are blacked out as well as branching information for the viewport configuration setup. It is sometimes necessary, under special circumstances, to be able to toggle bits on the timing and control board on the CIG. This task is also accomplished via the `view_flags` message.

2.1.2.2.4.1 clr_br_bit.c

(./simnet/release/src/vehicle/libsrc/libvflags/cclr_br_bit.c)

This file contains a routine which clears bits in a word in the *branch_values* array.

Includes:

```
"sim_types.h"
"mass_std.h"
"dgi_std.h"
"sim_cig_if.h"
"libvflags.h"
"vflags_loc.h"
```

2.1.2.2.4.1.1 clr_br_bit

This routine clears bits specified by *br_mask*. The element in the array is specified by *br_index*. The CIG (CIG 1 or CIG 2) is represented by *cig_index*.

Parameters		
Parameter	Type	Where Typedef Declared
br_index	int	Standard
br_mask	WORD	mass_std.h
cig_index	int	Standard

Table 2.1-306: `clr_br_bit` Information.

2.1.2.2.4.2 clr_vflags.c**(./simnet/release/src/vehicle/libsrc/libvflags/cclr_vflags.c)**

This file contains a routine which clears bits in a word in the *view_flags* array.

Includes:

"sim_types.h"
"mass_std.h"
"dgi_std.h"
"sim_cig_if.h"
"vflags_loc.h"

2.1.2.2.4.2.1 clear_view_flags

The bits specified by *flag_mask* in a word in the *view_flags* array are cleared. The CIG is specified by *index*.

Parameters		
Parameter	Type	Where Typedef Declared
flag_mask	WORD	mass_std.h
index	int	Standard

Table 2.1-307: clear_view_flags Information.

2.1.2.2.4.3 get_br_vals.c

(./simnet/release/src/vehicle/libsrc/libvflags/get_br_vals.c)

This file contains a routine which returns a pointer to the beginning of the *branch_values* array.

Includes:

"sim_types.h"
"mass_std.h"
"dgi_stdg.h"
"sim_cig_if.h"
"vflags_loc.h"

2.1.2.2.4.3.1 get_br_vals

A pointer to the beginning of the *branch_values* array is returned. The CIG buffer is specified by *index*.

Parameters		
Parameter	Type	Where Typedef Declared
index	int	Standard
Return Values		
Return Value	Type	Meaning
branch_values[index]	WORD	pointer to the beginning of the branch values array

Table 2.1-308: get_br_vals Information.

2.1.2.2.4.4 get_vflags.c

(./simnet/release/src/vehicle/libsrc/libvflags/get_vflags.c)

This file contains a routine which returns the value of a particular *view_flags* word.

Includes:

"sim_types.h"
"mass_stdh.h"
"dgi_stdh.h"
"sim_cig_if.h"
"vflags_loc.h"

2.1.2.2.4.4.1 get_view_flags

The value of a *view_flags* word is returned for the CIG specified by *index*.

Parameters		
Parameter	Type	Where Typedef Declared
index	int	Standard
Return Values		
Return Value	Type	Meaning
view_flags[index]	WORD	value of a view_flags word

Table 2.1-309: get_view_flags Information.

2.1.2.2.4.5 get_vmodes.c

(./simnet/release/src/vehicle/libsrc/libvflags/get_vmodes.c)

This file contains a routine which returns a pointer to the beginning of the *view_modes* array.

Includes:

```
"sim_types.h"
"mass_std.h"
"dgi_std.h"
"sim_cig_if.h"
"vflags_loc.h"
```

2.1.2.2.4.5.1 get_vmodes

A pointer to the beginning of the *view_modes* array is returned by this routine for the appropriate CIG, as indicated by *index*.

Parameters		
Parameter	Type	Where Typedef Declared
index	int	Standard
Return Values		
Return Value	Type	Meaning
view_modes[index]	pointer to WORD	pointer to the beginning of the <i>view modes</i> array

Table 2.1-310: get_vmodes Information.

2.1.2.2.4.6 set_br_bit.c

(./simnet/release/src/vehicle/libsrc/libvflags/set_br_bit.c)

This file contains a routine which sets bits in a word in the *branch_values* array.

Includes:

```
"sim_types.h"
"mass_std.h"
"dgi_std.h"
"sim_cig_if.h"
"vflags_loc.h"
```

2.1.2.2.4.6.1 set_br_bit

Bits specified by *br_mask* in a word in the *branch_values* array are set. The array element is specified by *br_index*, and the CIG is specified by *cig_index*.

Parameters		
Parameter	Type	Where Typedef Declared
<i>br_index</i>	int	Standard
<i>br_mask</i>	WORD	mass_std.h
<i>cig_index</i>	int	Standard

Table 2.1-311: set_br_bit Information.

2.1.2.2.4.7 set_br_vals.c

(./simnet/release/src/vehicle/libsrc/libvflags/set_br_vals.c)

This file contains a routine which sets the value of a word in the *branch_values* array.

Includes:

```
"sim_types.h"
"mass_std.h"
"dgi_std.h"
"sim_cig_if.h"
"vflags_loc.h"
```

2.1.2.2.4.7.1 set_br_vals

This routine sets the value of a word in the *branch_values* array to *br_mask*. The element in the array is specified by *br_index*. The appropriate CIG is represented by *cig_index*.

Parameters		
Parameter	Type	Where Typedef Declared
<i>br_index</i>	int	Standard
<i>br_mask</i>	WORD	mass_std.h
<i>cig_index</i>	int	Standard

Table 2.1-312: set_br_vals Information.

2.1.2.2.4.8 set_vflags.c

(/simnet/release/src/vehicle/libsrc/libvflags/set_vflags.c)

This file contains a routine which sets the bits in a word in the *view_flags* array.

Includes:

```
"sim_types.h"
"mass_std.h"
"dgi_std.h"
"sim_cig_if.h"
"vflags_loc.h"
```

2.1.2.2.4.8.1 set_view_flags

The bits specified by *flag_mask* in a word in the *view_flags* array are set. The CIG is specified by *index*.

Parameters		
Parameter	Type	Where Typedef Declared
flag_mask	WORD	mass_std.h
index	int	Standard

Table 2.1-313: set_view_flags Information.

2.1.2.2.4.9 set_vmodes.c

(/simnet/release/src/vehicle/libsrc/libvflags/set_vmodes.c)

The routine in this file sets the value of a word in the *view_modes* array.

Includes:

```
"sim_types.h"
"mass_std.h"
"dgi_std.h"
"sim_cig_if.h"
"vflags_loc.h"
```

2.1.2.2.4.9.1 set_vmodes

The value of a word in the *view_modes* array (*vm_index*) is set to *vm_mask*. The appropriate CIG is specified by *cig_index*.

Parameters		
Parameter	Type	Where Typedef Declared
vm_index	int	Standard
vm_mask	WORD	mass_std.h
cig_index	int	Standard

Table 2.1-314: set_vmodes Information.

2.1.2.2.4.10 vflags_loc.c

(./simnet/release/src/vehicle/libsrc/libvflags/vflags_loc.c)

The *view_flags*, *view_modes*, and *branch_values* arrays are declared in this file.

Includes:

```
"sim_types.h"  
"mass_stdh.h"  
"dgi_stdg.h"  
"sim_cig_if.h"  
"vflags_loc.h"
```

2.1.2.2.4.11 vision.c

(./simnet/release/src/vehicle/libsrc/libvflags/vision.c)

The routines in this file are not used by the Masscomp or Butterfly.

2.1.2.2.4.12 vflags_loc.h

(./simnet/release/src/vehicle/libsrc/libvflags/vflags_loc.h)

Declarations:

```
BRVALS[MAX_BRANCH_VALUES]  
VMODES[16]  
view_flags[]  
branch_values[]  
view_modes[]
```

2.1.2.2.4.13 libvflags.h

(./simnet/release/src/vehicle/libsrc/libvflags/libvflags.h)

The following constants are defined:

```
VIEWP_0  
VIEWP_1  
VIEWP_2  
VIEWP_3  
VIEWP_4  
VIEWP_5  
VIEWP_6  
VIEWP_7  
ALL_VIEWPORTS
```

The following functions are declared:

```
set_view_flags()  
clear_view_flags()  
get_view_flags()  
set_br_vals()  
get_br_vals()  
set_vmodes()  
get_vmodes()
```

2.1.2.2.5 libio_simul**(./simnet/release/src/libsrc/libio_simul [libio_simul])**

Libio_simul is instrumental in the sending and receiving of data message buffers. This library controls the input to and output from the CIG.

2.1.2.2.5.1 io_simul.c**(./simnet/release/src/libsrc/libio_simul/io_simul.c)**

Includes:

- "stdio.h"
- "sim_dfns.h"
- "sim_types.h"
- "mass_stdc.h"
- "dgi_stdg.h"
- "sim_cig_if.h"
- "rtc.h"
- "bbd.h"
- "libcig.h"
- "librva.h"
- "libnetwork.h"
- "status.h"
- "gbuffer.h"
- "libcig.h"

Declarations:

- last_time
- using_ethernet

2.1.2.2.5.1.1 io_simul

This routine controls the input to and output from the CIG during the simulation state. The routine first waits to receive a buffer from the CIG. To increase productivity while waiting, the routine processes packets from the network. After successfully receiving a buffer, the routine sends a message buffer to the CIG. The routine **check_buffer()** is called to check that the sent buffer contained valid messages. The routine also checks buffers sent to a second CIG (if present). The buffer received from the CIG is processed by calling **cig_process_buffer()**. Note that the CIG determines the start of a frame by sending another buffer to the host. **io_simul()** waits for the start of the next frame by processing network packets and polling the CIG until the next buffer is received.

Internal Variables		
Variable	Type	Where Typedef Declared
fail time	register long	Standard
Calls		
Function	Where Described	
net current time	See MCC CSCI SDD Section 2.20.2.8.3	
network get net handle	Section 2.1.1.3.2.12.1	
rtc start time	Section 2.6.16.1.2	
cig receive buffer	Section 2.1.2.2.1.9.1	
cig failed fsm	Sections 2.1.5.2, 2.1.5.3, and 2.1.5.4	
process a packet	Section 2.1.1.3.2.18.8	
cig send buffer	Section 2.1.2.2.1.10.1	
check buffer	Section 2.1.2.2.2.18.1	
get front of send buffer	Section 2.1.2.2.2.32.1	
get cig2 present	Section 2.1.2.2.1.17	
bbd bit out	Section 2.1.5.1.4.1	
cig process buffer	Section 2.1.2.2.1.7.1	
rtc stop time	Section 2.6.16.1.3	

Table 2.1-315: io_simul Information.

2.1.2.2.5.1.2 io_simul_idle

During idle state, it is still necessary for the simulation to communicate with the CIG. This routine controls the input to and output from the CIG during the idle state. As when the simulation is running, the routine waits to receive a buffer from the CIG. By calling the routine `network_xmit_idle()`, the routine is allowed to process packets from the network during the idle state. Once the CIG buffer is successfully received, the routine prepares and sends a null buffer to the CIG. As with the routine `io_simul()`, this routine waits to receive the next buffer from the CIG by polling the CIG and processing network packets. Note that during the idle state, the frame interruption will not necessarily come at regular 1/15 second intervals. The routine `buffer_reset()` is called in order to clear out the CIG buffer and reset the pointers.

Internal Variables		
Variable	Type	Where Typedef Declared
done	int	Standard
fail_time	register long	Standard
now	register long	Standard
Calls		
Function	Where Described	
cig_receive_buffer	Section 2.1.2.2.1.9.1	
net_current_time	See MCC CSCI SDD Section 2.20.2.8.3	
network_get_net_handle	Section 2.1.1.3.2.12.1	
cig_failed_fsm	Sections 2.1.5.2, 2.1.5.3, and 2.1.5.4	
cig_prepare_no_op	Section 2.1.2.2.1.4.1	
cig_send_buffer	Section 2.1.2.2.1.10.1	
network_xmit_idle	Section 2.1.1.3.1.32.3	
process_a_packet	Section 2.1.1.3.2.18.8	
buffer_reset	Section 2.1.2.2.2.15.1	

Table 2.1-316: io_simul_idle Information.

2.1.2.2.6 m1_cig.c

(./simnet/release/src/vehicle/m1/src/m1_cig.c [m1_cig.c])

The file m1_cig.c contains information regarding the CIG that is specific to the M1 simulator. It consists mainly of information about the specific vehicle's orientation and position, initialization messages specific to the vehicle, and the routine to actually format the buffer to be sent to the CIG.

Includes:

- "stdio.h"
- "signal.h"
- "math.h"
- "sim_types.h"
- "sim_dfns.h"
- "sim_macros.h"
- "mass_stdc.h"
- "dgi_stdg.h"
- "sim_cig_if.h"
- "libmsg.h"
- "pro_sim.h"
- "mun_type.h"
- "libkin.h"
- "librva.h"
- "libimps.h"
- "libhull.h"
- "libnetwork.h"
- "libturret.h"
- "gcom.h"
- "gbuffer.h"
- "libvflags.h"
- "m1_turret.h"
- "m1_vision.h"

Declarations:

- t_to_h_in_h[3]
- c_to_t_in_t[3]
- l_to_t_in_t[3]
- g_to_t-in_t[3]
- init_laser_ctr
- ball_debug
- cws_cos
- cws_sin
- lpscope_cos
- lpscope_sin
- tmp_mtx

Definitions:

GNR_3x
 GNR_10x
 GNR_MAG_BR_VAL
 H_TO_W_NODE
 T_TO_H_NODE
 G_TO_T_NODE
 LDR_PSCOPE
 CMDR_CUPOLA

2.1.2.2.6.1 set_ballistics_debug

This routine sets *ball_debug* to *state*.

Parameters		
Parameter	Type	Where Typedef Declared
state	int	standard

Table 2.1-317: set_ballistics_debug Information.

2.1.2.2.6.2 get_ballistics_debug

This routine returns *ball_debug*, indicating if debugging is on or off.

Return Values		
Return Value	Type	Meaning
ball_debug	int	TRUE - debugging is enabled FALSE - debugging is disabled

Table 2.1-318: get_ballistics_debug Information.

2.1.2.2.6.3 cig_init_ctr

This routine initializes a counter. It sets *init_laser_ctr* to 0.

2.1.2.2.6.4 cig_gps_mag_10x

This routine sets the magnification of the gunner's sight to 10x.

Calls	
Function	Where Described
set_br_vals	Section 2.1.2.2.4.7.1

Table 2.1-319: cig_gps_mag_10x Information.

2.1.2.2.6.4 cig_gps_mag_3x

This routine sets the magnification of the gunner's sight to 3x.

Calls	
Function	Where Described
set_br_vals	Section 2.1.2.2.4.7.1

Table 2.1-320: cig_gps_mag_3x Information.

2.1.2.2.6.5 cig_msg_prepend_my_veh_state

This routine prepends to the front of the CIG message buffer information necessary to correctly portray the host vehicle's orientation and position. The message which is sent to the graphics box to inform it of what to display is usually partially constructed. The information concerning the remote vehicles which are visible is laid out in the middle of the buffer; additional information is prepended above this part of the buffer. Most of the information in the buffer is related to the remote vehicles. Only information necessary to indicate a change in the host vehicle's position, orientation, and/or appearance is updated as required; therefore, less information is added to the buffer at each tick.

Internal Variables		
Internal Variable	Type	Where Typedef Declared
mat temp	T MATRIX	sim types.h
vec tmp	VECTOR	sim types.h
Calls		
Function	Where Described	
multi_cig_prepend_rts4x3_matrix	Section 2.1.2.2.2.82.1	
kinematics_get_w_to_h	Section 2.5.8.2.1	
kinematics_get_h_to_o	Section 2.5.8.2.3	
vec_mat_mul	Section 2.6.2.56.1	
mat_trig_init	Section 2.6.2.34.1	
multi_cig_prepend_viewflags	Section 2.1.2.2.2.89.1	
get_view_flags	Section 2.1.2.2.4.3.1	
get_br_vals	Section 2.1.2.2.4.4.1	
get_cig2_present	Section 2.1.2.2.1.17	
multi_cig_msg_prepend_request_laser_range	Section 2.1.2.2.2.81.1	

Table 2.1-321: cig_msg_prepend_my_veh_state Information.

2.1.2.2.6.6 cig_prepare_buffer

This routine prepares the CIG buffer. The routine `impacts_tell_cig_about_impacts()` is called to obtain special effects information for impacts. The routine `rva_tell_cig_about_other_vehicles()` is called to determine which vehicles are visible. Information is added to the buffer for new moving visible vehicles and is deleted from the buffer for former moving visible vehicles. This routine then calls

cig_adjust_for_changed_staticveh() to handle static vehicles that have changed vehicle type. The routine **cig_msg_prepend_staticveh_state()** is called to inform the CIG of new static vehicles that are to be displayed. The routine **cig_msg_prepend_staticveh_rem()** is called to inform the CIG of static vehicles that no longer need to be displayed.

Internal Variables		
Internal Variable	Type	Where Typedef Declared
num_static	int	standard
num_rm	int	standard
num_chg	int	standard
chgvehs	pointer to pointer to RVA_ENTRY	librva.h
staticvehs	pointer to pointer to RVA_ENTRY	librva.h
rmvehs	pointer to pointer to RVA_ENTRY	librva.h
sbp	pointer to S_BUFFER	libmsg.h
Calls		
Function	Where Described	
cig_not_ok_to_prepare_buffer	Section 2.1.2.2.1.23.1	
copy ballistics buffer	Section 2.1.2.2.2.14.1	
cig_msg_prepend_my_veh_state	Section 2.1.2.2.8.3	
copy Z rot to TF2	Section 2.6.4.9.1	
cig_msg_prepend_gun_overlay	Section 2.1.2.2.2.71.1	
firectl ready to fire	Section 2.2.2.2.3	
firectl malfunction	Section 2.2.2.2.3	
laser multiple returns	Section 2.2.3.2	
bcs get range str	Section 2.2.3.1	
turret get azimuth str	Section 2.5.5.2.19	
impacts_tell_cig_about_impacts	Section 2.5.15.1.2	
rva_tell_cig_about_other_vehicles	Section 2.5.12.27.1	
rva_get_lists	Section 2.5.12.9.2	
cig_adjust_for_changed_staticveh	Section 2.1.2.2.2.2.3	
cig_msg_prepend_staticveh_state	Section 2.1.2.2.2.98.1	
cig_msg_prepend_staticveh_rem	Section 2.1.2.2.2.97.1	

Table 2.1-322: cig_prepare_buffer Information.

2.1.2.2.6.7 cig_spec_init

This routine is called by libcig to initialize CIG-related information that is specific to a particular simulation. In the case of the M1, it requests that laser range values be returned every frame.

Calls	
Function	Where Described
multi_cig_prepend_request_laser_range	Section 2.1.2.2.81.1

Table 2.1-323: cig_spec_init Information.

2.1.2.2.6.8 cig_setup_configuration

This routine sets up the configuration file by calling the routine **cig_set_view_config_file()**. The configuration message is prepared via **cig_message_configure_view()**, and is sent to the CIG via **cig_send_buffer()**. Upon receipt of the configuration message, the CIG sends back an acknowledgement via **cig_receive_buffer()**.

Calls	
Function	Where Described
cig_set_view_config_file	Section 2.1.2.2.23.1
get_vconfig_file1	Section 2.5.1.2.2
get_cig2_present	Section 2.1.2.2.1.17.1
cig_send_buffer	Section 2.1.2.2.1.10.1
cig_receive_buffer	Section 2.1.2.2.1.9.1
get_vconfig_file2	Section 2.5.1.2.3
cig_msg_configure_view	Section 2.1.2.2.23.4

Table 2.1-324: cig_setup_configuration Information.

2.1.2.2.7 m2_cig.c

(./simnet/release/src/vehicle/m2/src/m2_cig.c [m2_cig.c])

The file m2_cig.c contains information regarding the CIG that is specific to the M2 simulator. It consists mainly of information about the specific vehicle's orientation and position, initialization messages specific to the vehicle, and the routine to actually format the buffer to be sent to the CIG.

Includes:

```
"stdio.h"
"signal.h"
"math.h"
"sim_dfns.h"
"sim_macros.h"
"sim_types.h"
"mass_stdh.h"
"dgi_stdg.h"
"sim_cig_if.h"
"libmsg.h"
"pro_sim.h"
"mun_type.h"
"libkin.h"
"librva.h"
"libimps.h"
"libhull.h"
"libturret.h"
"libnetwork.h"
"libmissile.h"
"libvflags.h"
"gcom.h"
"m2_ammo.h"
```

Declarations:

```
gps_hi_mag
mydebug
ball_debug
t_to_h_in_h[3]
c_to_t_in_t[3]
g_to_t_in_t[3]
brow_pad_first_time
tmp_m
cig_dead()
```

Definitions:

```
H_TO_W_NODE
T_TO_H_NODE
G_TO_T_NODE
GC_TO_T_NODE
C_TO_T_NODE
CR_TO_T_NODE
```

2.1.2.2.7.1 set_ballistics_debug

This routine sets *ball_debug* equal to *state*.

2.1.2.2.7.2 get_ballistics_debug

This routine returns the status of *ball_debug*, indicating whether or not debugging is enabled or disabled.

Return Values		
Return Value	Type	Meaning
ball_debug	int	status of ballistics debugging

Table 2.1-325: get_ballistics_debug Information.

2.1.2.2.7.3 init_brow_pad_state

brow_pad_first_time is initialized to 1.

2.1.2.2.7.4 cig_msg_prepend_my_veh_state

This routine prepends to the front of the CIG message buffer information necessary to correctly portray the host vehicle's orientation and position. The message which is sent to the graphics box to inform it of what to display is usually partially constructed. The information concerning the remote vehicles which are visible is laid out in the middle of the buffer; additional information is prepended above this part of the buffer. Most of the information in the buffer is related to the remote vehicles. Only information necessary to indicate a change in the host vehicle's position, orientation and/or appearance is updated as required; therefore, less information is added to the buffer at each tick.

Internal Variables		
Internal Variable	Type	Where Typedef Declared
mat temp	T MATRIX	sim types.h
vec tmp	VECTOR	sim types.h
cws sin	REAL	sim types.h
cws cos	REAL	sim types.h
Calls		
Function	Where Described	
cig_msg_prepend_rts4x3_matrix	Section 2.1.2.2.94.1	
kinematics_get_w_to_h	Section 2.5.8.2.1	
kinematics_get_h_to_o	Section 2.5.8.2.3	
vec_mat_mul	Section 2.6.2.56.1	
cupola_get_real_cws_sin_and_cos	Section 2.3.6.1.2.2	
mat_trig_init	Section 2.6.2.34.1	
cig_msg_prepend_viewflags	Section 2.1.2.2.106.1	
get_view_flags	Section 2.1.2.2.4.4.1	
get_br_vals	Section 2.1.2.2.4.3.1	
get_brow_pad_status	Section 2.3.6.3.2.27	
cig_msg_prepend_view_mode	Section 2.1.2.2.108.1	
get_vmodes	Section 2.1.2.2.4.5.1	

Table 2.1-326: cig_msg_prepend_my_veh_state Information.

2.1.2.2.7.5 cig_prepare_buffer

This routine prepares the CIG message buffer. The routine `impacts_tell_cig_about_impacts()` is called to obtain special effects information for impacts. The routine `rva_tell_cig_about_other_vehicles()` is called to determine which vehicles are visible. Information is added to the buffer for new moving visible vehicles and is deleted from the buffer for former moving visible vehicles. This routine then calls `cig_adjust_for_changed_staticveh()` to handle static vehicles that have changed. The routine `cig_msg_prepend_staticveh_state()` is called to inform the CIG of new static vehicles that are to be displayed. The routine `cig_msg_prepend_staticveh_rem()` is called to inform the CIG of static vehicles that no longer need to be displayed.

Internal Variables		
Internal Variable	Type	Where Typedef Declared
num_static	int	standard
num_rm	int	standard
num_chg	int	standard
chgvehs	pointer to pointer to RVA_ENTRY	librva.h
staticvehs	pointer to pointer to RVA_ENTRY	librva.h
rmvehs	pointer to pointer to RVA_ENTRY	librva.h
Calls		
Function	Where Described	
cig_not_ok_to_prepare_ buffer	Section 2.1.2.2.1.23.1	
copy_ballistics_buffer	Section 2.1.2.2.2.14.2	
cig_msg_prepend_my_veh_ state	Section 2.1.2.2.7.4	
cupola_get_cws_cos_and_ sin	Section 2.3.6.1.2.1	
cig_msg_prepend_gun_ overlay	Section 2.1.2.2.2.75.1	
turret_get_azimuth_str	Section 2.5.5.2.19	
bcs_get_range_str	Section 2.3.3.1.12	
map_get_ammo_entry_from_ network_type	Section 2.6.11.2.1	
ammo_round_indexed_status	Section 2.3.5.1.28	
impacts_tell_cig_about_ impacts	Section 2.5.15.1.2	
rva_tell_cig_about_other_ vehicles	Section 2.5.12.27.1	
rva_get_lists	Section 2.5.12.9.2	
cig_adjust_for_changed_ staticveh	Section 2.1.2.2.2.2.3	
cig_msg_prepend_staticveh_ state	Section 2.1.2.2.2.98.1	
cig_msg_prepend_staticveh_ rem	Section 2.1.2.2.2.97.1	

Table 2.1-327: cig-prepare_buffer Information.

2.1.2.2.7.6 cig_spec_init

This routine is called by libcig to initialize CIG-related information that is specific to a particular simulator. In the case of the M2, it prepends the ammo_define message.

Calls	
Function	Where Described
cig_msg_prepend_ammo_define	Section 2.1.2.2.67.1
map_get_ammo_entry_from_network_type	Section 2.6.11.2.1

Table 2.1-328: cig_spec_init Information.

2.1.2.2.8 kato_cig.c

(./simnet/release/src/vehicle/kato/src/kato_cig.c [kato_cig.c])

The file `kato_cig.c` contains information regarding the CIG that is specific to the Stealth vehicle simulation. It consists mainly of information about the specific vehicle's orientation and position, initialization messages specific to the vehicle, and the routine to actually format the buffer to be sent to the CIG.

Includes:

```
"stdio.h"
"math.h"
"sim_types.h"
"sim_dfns.h"
"sim_macros.h"
"mass_stdh.h"
"dgi_stdg.h"
"sim_cig_if.h"
"libmsg.h"
"pro_sim.h"
"mun_type.h"
"libkin.h"
"librva.h"
"libimps.h"
"libhull.h"
"libnetwork.h"
"libturret.h"
"librotate.h"
"libair.h"
"gbuffer.h"
"libvflags.h"
```

Declarations:

```
ident_mat
init_cig_ticks
```

Definition:

```
CIG_INIT_WAIT
```

2.1.2.2.8.1 cig_init_ctr

This routine initializes a counter. It sets `init_cig_ticks` to 0.

2.1.2.2.8.2 cig_local_init

This routine is a stub.

2.1.2.2.8.3 cig_msg_prepend_my_veh_state

This routine prepends to the front of the CIG message buffer information necessary to correctly portray the host vehicle's orientation and position. The message which is sent to the graphics box to inform the CIG of what to display is usually partially constructed. The information concerning the remote vehicles which are visible is laid out in the middle of the buffer; additional information is prepended above this part of the buffer. Most of the information in the buffer is related to the remote vehicles. Only information necessary to indicate a change in the host vehicle's position, orientation, and/or appearance is updated as required; therefore, less information is added to the buffer at each tick.

Calls	
Function	Where Described
rotate_send_msgs	Section 2.5.18.3.1
cig_msg_prepend_viewflags	Section 2.1.2.2.2.106.1
get_view_flags	Section 2.1.2.2.4.3.1
get_br_vals	Section 2.1.2.2.4.4.1

Table 2.1-329: cig_msg_prepend_my_veh_state Information.

2.1.2.2.8.4 cig_prepare_buffer

This routine prepares the CIG buffer. The routine `impacts_tell_cig_about_impacts()` is called to obtain special effects information for impacts. The routine `rva_tell_cig_about_other_vehicles()` is called to determine which vehicles are visible. Information is added to the buffer for new moving visible vehicles and is deleted from the buffer for former moving visible vehicles. This routine then calls `cig_adjust_for_changed_staticveh()` to handle static vehicles that have changed vehicle type. The routine `cig_msg_prepend_staticveh_state()` is called to inform the CIG of new static vehicles that are to be displayed. The routine `cig_msg_prepend_staticveh_rem()` is called to inform the CIG of static vehicles that no longer need to be displayed.

Internal Variables		
Internal Variable	Type	Where Typedef Declared
num_static	int	standard
num_rm	int	standard
num_chg	int	standard
chgvehs	pointer to pointer to RVA_ENTRY	librva.h
staticvehs	pointer to pointer to RVA_ENTRY	librva.h
rmvehs	pointer to pointer to RVA_ENTRY	librva.h
Calls		
Function	Where Described	
cig_not_ok_to_prepare_buffer	Section 2.1.2.2.1.23.1	
copy_ballistics_buffer	Section 2.1.2.2.2.14.1	
cig_msg_prepend_my_veh_state	Section 2.1.2.2.8.3	
impacts_tell_cig_about_impacts	Section 2.5.15.1.2	
rva_tell_cig_about_other_vehicles	Section 2.5.12.27.1	
rva_get_lists	Section 2.5.12.9.2	
cig_adjust_for_changed_staticveh	Section 2.1.2.2.2.2.3	
cig_msg_prepend_staticveh_state	Section 2.1.2.2.2.98.1	
cig_msg_prepend_staticveh_rem	Section 2.1.2.2.2.97.1	

Table 2.1-330: cig_prepare_buffer Information.

2.1.2.2.8.5 cig_spec_init

This routine is called by libcig to initialize CIG-related information that is specific to a particular simulation. In the case of the Stealth, it requests that an AGL value be returned every frame.

Calls	
Function	Where Described
cig_msp_prepend_agl_setup	Section 2.1.2.2.66.1

Table 2.1-331: cig_spec_init Information.

2.1.2.2.9 kato_view.c

(./simnet/release/src/vehicle/kato/src/kato_view.c [kato_view.c])

Kato_view.c allows the Stealth to view the battlefield.

Includes:

```
"stdio.h"
"sim_types.h"
"sim_dfns.h"
"sim_macros.h"
"libmsg.h"
"libmath.h"
"kato_cntrl.h"
"kato_view.h"
```

Defines:

<u>View Constant</u>	<u>Value</u>
VIEW_CENTERED	0.0
VIEW_MAX_SLEW	0.067
VIEW_PITCH_SLEW_PARAM	0.00009
VIEW_IND_DELTA	0.1745
VIEW_IND_NUM_SEG_ABOVE	4
VIEW_IND_NUM_SEG_BELOW	5
VIEW_MAG_NODE	4
VIEW_PITCH_NODE	3
VIEW_YAW_NODE	2
VIEW_SLEW_UP	0
VIEW_SLEW_DOWN	1
VIEW_SLEW_HOLD_PITCH	2
VIEW_SLEW_CENTER_PITCH	3
VIEW_SLEW_SET_PITCH	4
VIEW_SLEW_HOLD_YAW	5
VIEW_SLEW_CENTER_YAW	6
NLOS_VIEW_OFFSET	0.1
VIEW_NLOS_MAX_PITCH_ANGLE	0.7071
VIEW_NLOS_MAX_YAW_ANGLE	0.34906585

REAL variable declarations and initialization:

```
old_view_pitch_pos
view_pitch_base
view_pitch_set_angle = 0.0
old_view_yaw_pos
view_yaw_base
view_cpo_elevate_rate = 0.0
view_func_args[9]
```

int variable declarations:

```
view_pitch_slew_state
view_pitch_slew_count
view_yaw_slew_state
view_yaw_slew_count
```

REAL procedure declarations:

yaw_filter()
pitch_filter()

Macros invoked:

ROTATE_ELEMENT_DEF(view_pitch_element)
ROTATE_ELEMENT_DEF(view_yaw_element)
ROTATE_ELEMENT_DEF(view_mag_element)

2.1.2.2.9.1 view

This routine returns the pointer &view_pitch_element.

Return Values		
Return Value	Type	Meaning
&view_pitch_element	pointer to ROTATE_ELEMENT	Normal end

Table 2.1-332: view Information.

2.1.2.2.9.2 view_init

This routine is the view initialization routine.

Internal Variables		
Internal Variable	Type	Where Typedef Declared
view_pitch_bottom	REAL	sim_types.h
Errors		
Error Name	Reason for Error	
view_init: bad cubic	A call to find_cubic_func was unsuccessful.	
Calls		
Function	Where Described	
rotate_init_element	Section 2.5.18.4.2	
hull	Section 2.5.18.3.6	
rotate_init_cig_element	Section 2.5.18.3.1	
controls_view_ind_init	Section 2.2.2	
controls_view_ind_centered	Section 2.2.2	
find_cubic_func	Section 2.6.1.2.1	

Table 2.1-333: view_init Information.

2.1.2.2.9.3 view_simul

This routine is the view simulation routine.

Internal Variables		
Internal Variable	Type	Where Typedef Declared
temp	REAL	sim_types.h
view_pitch_pos	REAL	sim_types.h
new_seg	int	Standard
old_seg	int	Standard
new_yaw_angle	REAL	sim_types.h
new_pitch_angle	REAL	sim_types.h
Calls		
Function	Where Described	
rotate set angle	Section 2.5.18.4.17	
rotate get angle	Section 2.5.18.4.31	
controls view ind up	Section 2.2.2	
controls view ind down	Section 2.2.2	
rotate set rate	Section 2.5.18.4.18	
rotate set angle	Section 2.5.18.4.17	

Table 2.1-334: view_simul Information.

2.1.2.2.9.4 view_set_cpo_elevate_rate

This routine sets view_cpo_elevate_rate equal to *new_rate*.

Parameters		
Parameter	Type	Where Typedef Declared
new_rate	REAL	sim_types.h

Table 2.1-335: view_set_cpo_elevate_rate Information.

2.1.2.2.9.5 view_set_pitch_rate

This routine sets the pitch rate.

Parameters		
Parameter	Type	Where Typedef Declared
new_rate	REAL	sim_types.h
Calls		
Function	Where Described	
rotate set rate	Section 2.5.18.4.18	

Table 2.1-336: view_set_pitch_rate Information.

2.1.2.2.9.6 view_centered

This routine centers the view.

Calls	
Function	Where Described
rotate_set_max_rate	Section 2.5.18.4.10

Table 2.1-337: view_centered Information.

2.1.2.2.9.7 view_up_depressed

Calls	
Function	Where Described
rotate_set_max_rate	Section 2.5.18.4.10

Table 2.1-338: view_up_depressed Information.

2.1.2.2.9.8 view_up_released

This routine sets view_pitch_slew_state equal to the defined constant VIEW_SLEW_HOLD_PITCH.

2.1.2.2.9.9 view_down_depressed

Calls	
Function	Where Described
rotate_set_max_rate	Section 2.5.18.4.10

Table 2.1-339: view_down_depressed Information.

2.1.2.2.9.10 view_down_released

This routine sets view_pitch_slew_state equal to the defined constant VIEW_SLEW_HOLD_PITCH.

2.1.2.2.9.11 view_to_world

Before this routine, the two variables VIEW_C_WORLD and VIEW_C_BODY are declared as type T_MATRIX.

Return Values		
Return Value	Type	Meaning
rotate_get_mat(...)	T_MAT_PTR	Normal end
Calls		
Function	Where Described	
rotate_get_mat	Section 2.5.18.5.5	
world	Section 2.5.18.6.8	

Table 2.1-340: view_to_world Information.

2.1.2.2.9.12 view_get_desired_missile_heading

This routine has been commented out. It declares one internal variable.

Internal Variables		
Internal Variable	Type	Where Typedef Declared
new_ang	REAL	sim_types.h

Table 2.1-341: view_get_desired_missile_heading Information.

2.1.2.2.9.13 view_get_pitch_angle

This routine returns the pitch angle.

Internal Variables		
Internal Variable	Type	Where Typedef Declared
pitch_angle	REAL	sim_types.h
Return Values		
Return Value	Type	Meaning
pitch_angle	REAL	Normal end
Calls		
Function	Where Described	
rotate_get_angle	Section 2.5.18.4.31	

Table 2.1-342: view_get_pitch_angle Information.

2.1.2.2.9.14 view_get_yaw_angle

This routine returns the yaw angle.

Internal Variables		
Internal Variable	Type	Where Typedef Declared
yaw angle	REAL	sim_types.h
Return Values		
Return Value	Type	Meaning
yaw angle	REAL	Normal end
Calls		
Function	Where Described	
rotate get angle	Section 2.5.18.4.31	

Table 2.1-343: view_get_yaw_angle Information.

2.1.2.2.9.15 yaw_filter

Constants for use in the following routines are defined prior to this routine as follows:

Constant	Value
A	100.0
B	200.0
C	100.0
D	1500.0
E	-1800.0
F	300.0

Parameters		
Parameter	Type	Where Typedef Declared
angle	REAL	sim_types.h
Internal Variables		
Internal Variable	Type	Where Typedef Declared
theta_d[3]	double	Standard
theta[3]	double	Standard
Return Values		
Return Value	Type	Meaning
theta[0]	REAL	angle

Table 2.1-344: yaw_filter Information.

2.1.2.2.9.16 pitch_filter

Parameters		
Parameter	Type	Where Typedef Declared
angle	REAL	sim_types.h
Internal Variables		
Internal Variable	Type	Where Typedef Declared
theta_d[3]	double	Standard
theta[3]	double	Standard
Return Values		
Return Value	Type	Meaning
theta[0]	REAL	angle

Table 2.1-345: pitch_filter Information.**2.1.2.2.9.17 view_set_pitch_angle**

This routine sets the pitch angle.

Parameters		
Parameter	Type	Where Typedef Declared
pitch	REAL	sim_types.h
Calls		
Function	Where Described	
rotate_set_max_rate	Section 2.5.18.4.10	

Table 2.1-346: view_set_pitch_angle Information.

2.1.3.1 libsound.c

(./simnet/release/src/libsrc/libsound.c [libsound.c])

Includes:

"stdio.h"
"math.h"
"sim_dfns.h"
"sim_macros.h"
"sim_types.h"
"basic.h"
"obj_type.h"
"timers_dfn.h"
"timers.h"
"libsound.h"
"libsound_dfn.h"
"libmap.h"
"libmap_dfn.h"

External procedure declarations:

sound_make_veh_spec_sound()
sound_force_veh_spec_sound()

Defines:

RANGE_SQ_CLOSE
RANGE_SQ_MED
RANGE_SQ_FAR
MAX_VAR_SOUND

2.1.3.1.1 sound_of_weapons_impact

The vehicle specific code contains an array of sounds to be made. This array classifies the different types of sounds in a consistent order for all types of vehicles based on the type of ammo, the type of impact, and the distance from the vehicle to the impact. This routine uses the parameters of a specific impact to calculate the index into the sound array which will determine which sound is to be made.

The input parameters are:

- ammo_index* -- the index into the ammo mapping array (describes the type of ammo)
- impact_type* -- the type of impact (muzzle flash, impact, no impact, etc.)
- range_sq* -- the distance from the vehicle to the impact, for determining sound delays

Parameters		
Parameter	Type	Where Typedef Declared
ammo_index	int	Standard
impact_type	int	Standard
range_sq	REAL	Standard
Internal Variables		
Variable	Type	Where Typedef Declared
sound_index	int	Standard
ammo_type	int	Standard
Calls		
Function	Where Described	
map_get_ammo_class_from_ammo_entry	Section 2.6.11.2.13	
timers_delay_proc	Section 2.6.3.4.1	
frame_delay_of_sound		
sound_make_const_sound	Section 2.1.3.1.2	

Table 2.1-347: sound_of_weapons_impact Information.

2.1.3.1.2 sound_make_const_sound

This routine determines which constant sounds are to be made, and calls the vehicle specific code to interface with the sound system. The parameter *sound_index* is an index into the sound array in the vehicle specific code. Note that the vehicle specific code actually makes the sound with the string passed by this routine.

Parameters		
Parameter	Type	Where Typedef Declared
sound_index	int	Standard
Internal Variables		
Variable	Type	Where Typedef Declared
length	int	Standard
Calls		
Function	Where Described	
sound make veh spec sound	Sections 2.1.3.2.2, 2.1.3.3.2, and 2.1.3.4.2	

Table 2.1-348: sound_make_const_sound Information.

2.1.3.1.3 sound_force_const_sound

This routine forces a constant sound specified in the parameter *sound_index*, regardless of whether the vehicle specific code disallows the sound.

Parameters		
Parameter	Type	Where Typedef Declared
sound_index	int	Standard
Internal Variables		
Variable	Type	Where Typedef Declared
length	int	Standard
Calls		
Function	Where Described	
sound force veh spec sound	Sections 2.1.3.2.3, 2.1.3.3.3, and 2.1.3.4.3	

Table 2.1-349: sound_force_const_sound Information.

2.1.3.1.4 sound_make_var_sound

This routine determines which variable sounds are to be made, and calls the vehicle specific code to interface with the sound system. An example of a variable sound is the engine noise, whose pitch varies with the speed of the vehicle and the terrain. The parameter *sound_index* is an index into the sound array in the vehicle specific code. *pct* is the percent of the pitch range at which the sound is to be made. Note that the vehicle specific code actually makes the sound with the string passed by this routine.

Parameters		
Parameter	Type	Where Typedef Declared
sound_index	int	Standard
pct	REAL	Standard
Internal Variables		
Variable	Type	Where Typedef Declared
mod	int	Standard
buf[10]	char	Standard
Return Values		
Return Value	Type	Meaning
TRUE	int	the sound index is valid
FALSE	int	the sound index is invalid
Calls		
Function	Where Described	
sound make veh spec sound	Sections 2.1.3.2.2, 2.1.3.3.2, and 2.1.3.4.2	

Table 2.1-350: sound_make_var_sound Information.

2.1.3.1.5 sound_get_var_sound_arg

This routine determines the modifier, *mod*, that is used to vary the sound instead of a percent.

Parameters		
Parameter	Type	Where Typedef Declared
sound_index	int	Standard
pct	REAL	Standard
mod	pointer to int	Standard
Return Values		
Return Value	Type	Meaning
veh_sound_array[sound_index].last_mod != *mod	int	modifier to send to the sound system

Table 2.1-351: sound_get_var_sound_arg Information.

2.1.3.1.6 sound_make_arg_sound

This routine determines the variable sound to be made given the index into the sound array, *sound_index*, and the modifier, *mod*. The vehicle specific code is called to interface with the sound system.

Parameters		
Parameter	Type	Where Typedef Declared
<i>sound_index</i>	int	Standard
<i>mod</i>	int	Standard
Internal Variables		
Variable	Type	Where Typedef Declared
<i>buf[10]</i>	char	Standard
Calls		
Function	Where Described	
<i>sound_make veh spec sound</i>	Sections 2.1.3.2.2, 2.1.3.3.2, and 2.1.3.4.2	

Table 2.1-352: *sound_make_arg_sound* Information.

2.1.3.1.7 sound_make_del_sound

This routine makes a delayed sound. It has timers call this routine after a certain delay. The argument for this routine is passed by the timers library.

Calls	
Function	Where Described
<i>timers get data</i>	Section 2.6.3.3.1
<i>sound make const sound</i>	Section 2.1.3.1.2

Table 2.1-353: *sound_make_del_sound* Information.

2.1.3.1.8 sound_force_del_sound

This routine forces a delayed sound, regardless of the timing and of whether the vehicle specific code disallows the sound.

Calls	
Function	Where Described
<i>sound force const sound</i>	Section 2.1.3.1.3
<i>timers get data</i>	Section 2.6.3.3.1

Table 2.1-354: *sound_force_del_sound* Information.

2.1.3.1.9 sound_make_cont_sound

This routine combines different sounds into a continuous sound. For example, it ensures that the different components of the engine sound (startup sound, running sound, and stopping sound) are made without any breaks. *start_index* is the index into the sound array for the starting sound; *vary_index* is the index into the sound array for a variable sound; *stop_index* is the index into the sound array for the stopping sound; *pct* is the percent for varying the variable pitch sound.

Parameters		
Parameter	Type	Where Typedef Declared
start_index	int	Standard
vary_index	int	Standard
stop_index	int	Standard
pct	REAL	Standard
Internal Variables		
Variable	Type	Where Typedef Declared
mod	int	Standard
length	int	Standard
buff[10]	char	Standard
Return Values		
Return Value	Type	Meaning
FALSE	int	no sound was made
TRUE	int	a sound was made
Calls		
Function	Where Described	
sound_make_veh_spec_sound	Sections 2.1.3.2.2, 2.1.3.3.2, and 2.1.3.4.2	

Table 2.1-355: sound_make_cont_sound Information.

2.1.3.1.10 sound_stop_cont_sound

This routine stops a continuous sound.

Parameters		
Parameter	Type	Where Typedef Declared
stop_index	int	Standard
vary_index	int	Standard
Internal Variables		
Variable	Type	Where Typedef Declared
length	int	Standard
Calls		
Function	Where Described	
sound make veh spec sound	Sections 2.1.3.2.2, 2.1.3.3.2, and 2.1.3.4.2	

Table 2.1-356: sound_stop_cont_sound Information.

2.1.3.2 m1_sound.c

(./simnet/release/src/vehicle/m1/m1_sound.c [m1_sound.c])

Includes:

"stdio.h"	"sim_cig_if.h"
"ctype.h"	"libfail.h"
"signal.h"	"libidc_dfn.h"
"sim_dfns.h"	"libsound.h"
"sim_macros.h"	"libsound_dfn.h"
"fifo_dfn.h"	"m1_mem_dfn.h"
"fifo.h"	"m1_sound.h"
"mass_stdc.h"	"m1_sound_dfn.h"
"dgi_stdg.h"	"timers_dfn.h"
"timers.h"	

Defines:

TICKS_BETWEEN_RANDOM_SOUNDS
 MAX_NUMBER_OF_RANDOM_SOUNDS
 CAT_KILL_DELAY
 SOUND_PORT_NUM
 DRIVING_ON_ROAD_FACTOR
 NO_TRAVERSE
 ELECTRIC_TRAVERSE
 HYDRAULIC_TRAVERSE

Declarations:

veh_sound_array	-- The listing of all possible sounds the vehicle can make. Each sound is listed as an index into the array.
soil_type	-- The current soil type
turret_traverse_state	-- The turret traverse state
counter	-- A counter for timeouts on random sounds
sound_index	-- Pointer into the sound array
gun_is_elevating	-- Flag for elevation sounds
gun_is_hitting_stop	-- Flag for hitting stops
track_speed_factor	-- Fraction of max truck speed
dont_use_sound	-- Set to zero if sounds are to be used
sound_of_random_sounds()	
sound_denial_check()	
sound_array	-- The array of random sounds

2.1.3.2.1 sound_denial_check

This routine returns TRUE if the sound system is not used. Access to the sound system is denied if the *dont_use_sound* flag is equal to 1, if the vehicle was destroyed, or if *sim_state_sounds_denied()* returns FALSE. The *dont_use_sound* flag is set from the command line.

Return Values		
Return Value	Type	Meaning
TRUE	int	The use of sounds is denied
FALSE	int	The use of sounds is not denied
Calls		
Function	Where Described	
fail_death_status	Section 2.5.4.10.1	
sim_state_sounds_denied	Section 2.5.1.1.12	

Table 2.1-357: sound_denial_check Information.

2.1.3.2.2 sound_make_veh_spec_sound

This routine takes a sound request string, *sound_str*, of length *str_len* from the sound library and requests the sound from the sound system. The routine checks if access to the sound system is allowed, and queues the sound request to the fifo.

Parameters		
Parameter	Type	Where Typedef Declared
sound_str	array of char	Standard
str_len	int	Standard
Calls		
Function	Where Described	
sound_denial_check	Section 2.1.3.2.1	
fifo_enqueue	Section 2.6.8.2.1	

Table 2.1-358: sound_make_veh_spec_sound Information.

2.1.3.2.3 sound_force_veh_spec_sound

This routine takes a sound request, *sound_str*, from the sound library and requests the sound from the sound system. The routine checks if access to the sound system is allowed, and queues the sound request to the fifo. Access to the sound system may only be restricted from the command line.

Parameters		
Parameter	Type	Where Typedef Declared
<i>sound_str</i>	array of char	Standard
<i>str_len</i>	int	Standard
Calls		
Function	Where Described	
sound denial check	Section 2.1.3.2.1	
fifo enqueue	Section 2.6.8.2.1	

Table 2.1-359: sound_force_veh_spec_sound Information.

2.1.3.2.4 sound_init

This routine checks if access to the sound system is allowed, initializes the fifo, and resets the sound system.

Calls	
Function	Where Described
fifo init	Section 2.8.6.3.1
sound reset	Section 2.1.3.2.7

Table 2.1-360: sound_init Information.

2.1.3.2.5 sound_dont_use

This routine sets the *dont_use_sound* flag to 1 and prints the statement "Sound is turned off".

2.1.3.2.6 sound_simul

This routine runs the sound simulation. It checks if access to the sound system is allowed, queues sounds to the fifo, and calls for random sounds to be made.

Calls	
Function	Where Described
fifo enqueue	Section 2.8.6.2.1
sound of random sounds	Section 2.1.3.2.12

Table 2.1-361: sound_simul Information.

2.1.3.2.7 sound_reset

This routine checks if access to the sound system is allowed and resets the sound system.

Calls	
Function	Where Described
fifo_enqueue	Section 2.6.8.2.1

Table 2.1-362: sound_reset Information.

2.1.3.2.8 sound_we_just_died

This routine resets the sound system. At a later tick, the sound of catastrophic kill is requested.

Calls	
Function	Where Described
sound_reset	Section 2.1.3.2.7
timers_delay_proc	Section 2.6.3.4.1

Table 2.1-363: sound_we_just_died Information.

2.1.3.2.9 sound_of_tracks

This routine makes the sound of the tracks starting, stopping, and moving, based on the soil type and the track speed.

Parameters		
Parameter	Type	Where Typedef Declared
fraction_of_max_speed	REAL	/simnet/common/include /global/sim_types.h
new_soil_type	int	Standard
Internal Variables		
Variable	Type	Where Typedef Declared
temp_tracks_speed	int	Standard
sound_of_tread_start	int	Standard
sound_of_tread_stop	int	Standard
sound_of_vary_tread	int	Standard
sound_of_vary_other_tread	int	Standard
Calls		
Function	Where Described	
sound_get_var_sound_arg	Section 2.1.3.1.5	
sound_make_const_sound	Section 2.1.3.1.2	
sound_make_arg_sound	Section 2.1.3.1.6	

Table 2.1-364: sound_of_tracks Information.

2.1.3.2.10 sound_of_turret_traversing

This routine makes the sound of the turret starting, stopping and traversing, based on whether the turret is under hydraulic or electric traverse control.

Parameters		
Parameter	Type	Where Typedef Declared
traversing	int	Standard
palm_release_or_hydraulic	int	Standard
fraction	REAL	/simnet/common/include /global/sim_types.h
Internal Variables		
Variable	Type	Where Typedef Declared
new_traverse_state	register int	Standard
Calls		
Function	Where Described	
sound_make_const_sound	Section 2.1.3.1.2	
sound_make_var_sound	Section 2.1.3.1.4	

Table 2.1-365: sound_of_turret_traversing Information.

2.1.3.2.11 sound_of_gun_elevating

This routine makes the sounds of the gun starting to elevating, elevating, and hitting the elevation stops.

Parameters		
Parameter	Type	Where Typedef Declared
fraction_of_max_speed	REAL	/simnet/common/include /global/sim_types.h
temp_hitting_stop	int	Standard
Calls		
Function	Where Described	
sound_make_const_sound	Section 2.1.3.1.2	

Table 2.1-366: sound_of_gun_elevating Information.

2.1.3.2.12 sound_of_random_sounds

This routine makes the random squeak, rattle, and bump sounds.

Calls	
Function	Where Described
roll dice	/simnet/common/include/global/sim_macros.h
sound make const sound	Section 2.1.3.1.2

Table 2.1-367: sound_of_random_sounds Information.

2.1.3.3 m2_sound.c

(./simnet/release/src/vehicle/m2/m2_sound.c [m2_sound.c])

Includes:

"stdio.h"	"sim_cig_if.h"
"ctype.h"	"libfail.h"
"signal.h"	"libidc_dfn.h"
"sim_dfns.h"	"libsound.h"
"sim_macros.h"	"libsound_dfn.h"
"fifo_dfn.h"	"m2_mem_dfn.h"
"fifo.h"	"m2_sound.h"
"mass_std.c.h"	"m2_sound_dfn.h"
"dgi_stdg.h"	"timers_dfn.h"
"timers.h"	"sim_types.h"

Defines:

TICKS_BETWEEN_RANDOM_SOUNDS
 MAX_NUMBER_OF_RANDOM_SOUNDS
 CAT_KILL_DELAY
 SOUND_PORT_NUM
 DRIVING_ON_ROAD_FACTOR
 TRAVERSE_ABRUPT_STOP_SPEED
 MAIN_GUN_FIRING_DELAY
 FAST_TRAV_ABRUPT_STOP_DELAY
 FAST_TRAV_NORMAL_STOP_DELAY
 SLOW_TRAV_ABRUPT_STOP_DELAY
 SLOW_TRAV_NORMAL_STOP_DELAY
 TURRET_DRIVE_OFF_DELAY
 ENGINE_CRANKING_STOP_DELAY
 ENGINE_STALLING_DELAY
 ENGINE_RUNNING_STOP_DELAY
 TURRET_POWER_PRIORITY
 TURRET_DRIVE_PRIORITY
 TURRET_TRAVERSE_PRIORITY
 ENGINE_ACCESSORY_PRIORITY
 ENGINE_RUNNING_PRIORITY

Declarations:

veh_sound_array	-- The listing of all possible sounds the vehicle can make. Each sound is listed as an index into the array.
soil_type	-- The current soil type
counter	-- A counter for timeouts on random sounds
sound_index	-- Pointer into the sound array
gun_is_fast_elevating	-- Flag for fast elevate sounds
gun_is_slow_elevating	-- Flag for slow elevate sounds
gun_is_hitting_stops	-- Flag for hitting stops
turret_is_fast_traversing	-- Flag for fast traverse sounds
turret_is_slow_traversing	-- Flag for slow traverse sounds
old_traverse_speed	-- For traverse abrupt stop
track_speed_factor	-- Fraction of max track speed
main_gun_firing_counter	-- For choosing firing sound
turret_drive_should_be_on	For queueing on channel 2

```

turret_power_should_be_on  For queueing on channel 2
channel_2_priority          -- For queueing on channel 2
channel_2_counter           -- For queueing on channel 2
engine_accessory_should_be_on  For queueing on channel 5
channel_5_priority          -- For queueing on channel 5
channel_5_counter           -- For queueing on channel 5
dont_use_sound              -- Set to zero if sounds are to be used
sound_of_random_sounds()
sound_denial_check()
sound_of_turret_power_already_on()
sound_of_turret_drive_already_on()
sound_of_engine_accessory_already_on()
channel_2_check()
channel_5_check()
sound_array                 -- The array of random sounds

```

2.1.3.3.1 sound_denial_check

This routine returns TRUE if the sound system is not used. Access to the sound system is denied if the *dont_use_sound* flag is equal to 1, if the vehicle was destroyed, or if the *sim_state_sounds_denied()* returns FALSE. The *dont_use_sound* flag is set from the command line.

Return Values		
Return Value	Type	Meaning
TRUE	int	The use of sounds is denied
FALSE	int	The use of sounds is not denied
Calls		
Function	Where Described	
fail_death_status	Section 2.5.4.10.1	
sim_state_sounds_denied	Section 2.5.1.1.12	

Table 2.1-368: sound_denial_check Information.

2.1.3.3.2 sound_make_veh_spec_sound

This routine takes a sound request, *sound_str*, from the sound library and requests the sound from the sound system. The routine checks if access to the sound system is allowed, and queues the sound request to the fifo.

Parameters		
Parameter	Type	Where Typedef Declared
sound_str	array of char	Standard
str_len	int	Standard
Calls		
Function	Where Described	
sound_denial_check	Section 2.1.3.3.1	
fifo_enqueue	Section 2.6.8.2.1	

Table 2.1-369: sound_make_veh_spec_sound Information.

2.1.3.3.3 sound_force_veh_spec_sound

This routine takes a sound request, *sound_str*, from the sound library and requests the sound from the sound system. The routine checks if access to the sound system is allowed, and queues the sound request to the fifo. Access to the sound system may only be disallowed at the command line.

Parameters		
Parameter	Type	Where Typedef Declared
sound_str	array of char	Standard
str_len	int	Standard
Calls		
Function	Where Described	
sound_denial_check	Section 2.1.3.3.1	
fifo_enqueue	Section 2.6.8.2.1	

Table 2.1-370: sound_force_veh_spec_sound Information.

2.1.3.3.4 sound_init

This routine checks if access to the sound system is allowed, initializes the fifo, and resets the sound system.

Calls	
Function	Where Described
fifo_init	Section 2.6.8.3.1
sound_reset	Section 2.1.3.3.7

Table 2.1-371: sound_init Information.

2.1.3.3.5 sound_dont_use

This routine sets the *dont_use_sound* flag to 1 and prints the statement "Sound is turned off".

2.1.3.3.6 sound_simul

This routine runs the sound simulation. It checks if access to the sound system is allowed, queues sounds to the fifo, calls for random sounds to be made, monitors the channel 2 sound port and the channel 5 sound port, and decrements the main gun firing counter.

Calls	
Function	Where Described
fifo_enqueue	Section 2.6.8.2.1
sound_of_random_sounds	Section 2.1.3.3.7
channel 2 check	Section 2.1.3.3.29
channel 5 check	Section 2.1.3.3.30

Table 2.1-372: sound_simul Information.

2.1.3.3.7 sound_reset

This routine checks if access to the sound system is allowed and resets the sound system.

Calls	
Function	Where Described
fifo_enqueue	Section 2.6.8.2.1

Table 2.1-373: sound_reset Information.

2.1.3.3.8 sound_we_just_died

This routine resets the sound system. At a later tick, the sound of catastrophic kill is requested.

Calls	
Function	Where Described
sound_reset	Section 2.1.3.3.7
timers_delay_proc	Section 2.6.3.4.1

Table 2.1-374: sound_we_just_died Information.

2.1.3.3.9 sound_of_main_gun_firing

This routine checks if access to the sound system is allowed, and makes the sound of the main gun firing. The number of times the sound is made is based on the status of the main gun firing counter, which depends on whether the gun is in continuous or single shot mode.

Calls	
Function	Where Described
sound denial check	Section 2.1.3.3.1
sound make const sound	Section 2.1.3.1.2

Table 2.1-375: sound_of_main_gun_firing Information.

2.1.3.3.10 sound_of_engine_cranking_start

This routine checks if access to the sound system is allowed, and makes the sound of the engine cranking being started. The sound of the start of the engine cranking is given high priority on Channel 5.

Calls	
Function	Where Described
sound denial check	Section 2.1.3.3.1
sound make const sound	Section 2.1.3.1.2

Table 2.1-376: sound_of_engine_cranking_start Information.

2.1.3.3.11 sound_of_engine_cranking_stop

This routine checks if access to the sound system is allowed, and makes the sound of the engine cranking being stopped. The sound of the engine cranking stop is given high priority on Channel 5.

Calls	
Function	Where Described
sound denial check	Section 2.1.3.3.1
sound make const sound	Section 2.1.3.1.2

Table 2.1-377: sound_of_engine_cranking_stop Information.

2.1.3.3.12 sound_of_engine_cranking_stall

This routine checks if access to the sound system is allowed, and makes the sound of the engine cranking stalling. The sound of the engine cranking stall is given high priority on Channel 5.

Calls	
Function	Where Described
sound denial check	Section 2.1.3.3.1
sound make const sound	Section 2.1.3.1.2

Table 2.1-378: sound_of_engine_cranking_stall Information.

2.1.3.3.13 sound_of_tracks

This routine makes the sound of the tracks starting, stopping, and moving, based on the soil type and the vehicle speed.

Parameters		
Parameter	Type	Where Typedef Declared
fraction_of_max_speed	REAL	/simnet/common/include /global/sim_types.h
new soil type	int	Standard
Internal Variables		
Variable	Type	Where Typedef Declared
temp tracks speed	int	Standard
sound of tread start	int	Standard
sound of tread stop	int	Standard
sound of vary tread	int	Standard
sound of vary other tread	int	Standard
Calls		
Function	Where Described	
sound get var sound arg	Section 2.1.3.1.5	
sound make const sound	Section 2.1.3.1.3	
sound make arg sound	Section 2.1.3.1.6	

Table 2.1-379: sound_of_tracks Information.

2.1.3.3.14 sound_of_engine_start

This routine checks if access to the sound system is allowed, and makes the sound of the engine starting. The sound of the engine start is given a high priority on Channel 5.

Calls	
Function	Where Described
sound denial check	Section 2.1.3.3.1
sound make const sound	Section 2.1.3.1.2

Table 2.1-380: sound_of_engine_start Information.

2.1.3.3.15 sound_of_engine_stop

This routine checks if access to the sound system is allowed, and makes the sound of the engine stopping. The sound of the engine stop is given a high priority on Channel 5.

Calls	
Function	Where Described
sound denial check	Section 2.1.3.3.1
sound make const sound	Section 2.1.3.1.2

Table 2.1-381: sound_of_engine_stop Information.

2.1.3.3.16 sound_of_engine

This routine makes the sound of the engine.

Parameters		
Parameter	Type	Where Typedef Declared
fraction_of_max_speed	REAL	/simnet/common/include /global/sim_types.h
Calls		
Function	Where Described	
sound make const sound	Section 2.1.3.1.2	

Table 2.1-382: sound_of_engine Information.

2.1.3.3.17 sound_of_gun_elevating

This routine makes the sounds of the gun elevating and hitting the elevation stops, based on whether the gun is elevating quickly or slowly.

Parameters		
Parameter	Type	Where Typedef Declared
fraction_of_max_speed	REAL	/simnet/common/include/global/sim_types.h
temp_hitting_stop	int	Standard
temp_fast	BOOLEAN	/simnet/common/include/global/sim_types.h
Calls		
Function	Where Described	
sound_make_const_sound	Section 2.1.3.1.2	

Table 2.1-383: sound_of_gun_elevating Information.

2.1.3.3.18 sound_of_turret_traversing

This routine checks if access to the sound system is allowed, and makes the sound of the turret traversing and stopping, based on whether the turret is moving slowly or quickly and whether the stop is normal or abrupt. The priority level of Channel 2 is checked before making the sound.

Parameters		
Parameter	Type	Where Typedef Declared
fraction_of_max_speed	REAL	/simnet/common/include/global/sim_types.h
temp_fast	BOOLEAN	/simnet/common/include/global/sim_types.h
Calls		
Function	Where Described	
sound_denial_check	Section 2.1.3.3.1	
fifo_enqueue	Section 2.6.8.2.1	
sound_make_const_sound	Section 2.1.3.1.2	

Table 2.1-384: sound_of_turret_traversing Information.

2.1.3.3.19 sound_of_turret_power_on

This routine checks if access to the sound system is allowed, and makes the sound of the turret power being on. Note that the turret power sound, turret drive sound, and turret traverse sound are all on Channel 2 with different priorities. The low priority sounds will be turned on once the higher priority sounds are turned off.

Calls	
Function	Where Described
sound denial check	Section 2.1.3.3.1
sound make const sound	Section 2.1.3.1.2

Table 2.1-385: sound_of_turret_power_on Information.

2.1.3.3.20 sound_of_turret_power_already_on

This routine checks if access to the sound system is allowed, and makes the sound of the turret power already being on. The priority level of Channel 2 is checked before making the sound.

Calls	
Function	Where Described
sound denial check	Section 2.1.3.3.1
sound make const sound	Section 2.1.3.1.2

Table 2.1-386: sound_of_turret_power_already_on Information.

2.1.3.3.21 sound_of_turret_power_off

This routine checks if access to the sound system is allowed, and makes the sound of the turret power being off. The priority level of Channel 2 is checked before making the sound.

Calls	
Function	Where Described
sound denial check	Section 2.1.3.3.1
sound make const sound	Section 2.1.3.1.2

Table 2.1-387: sound_of_turret_power_off Information.

2.1.3.3.22 sound_of_turret_drive_on

This routine checks if access to the sound system is allowed, and makes the sound of the turret drive being turned on. The priority level of Channel 2 is checked before making the sound.

Calls	
Function	Where Described
sound denial check	Section 2.1.3.3.1
sound make const sound	Section 2.1.3.1.2

Table 2.1-388: sound_of_turret_drive_on Information.

2.1.3.3.23 sound_of_turret_drive_already_on

This routine checks if access to the sound system is allowed, and makes the sound of the turret drive already being on. The priority level of Channel 2 is checked before making the sound.

Calls	
Function	Where Described
sound denial check	Section 2.1.3.3.1
sound make const sound	Section 2.1.3.1.2

Table 2.1-389: sound_of_turret_drive_already_on Information.

2.1.3.3.24 sound_of_turret_drive_off

This routine checks if access to the sound system is allowed, and makes the sound of the turret drive being turned off. The priority level of Channel 2 is checked before making the sound.

Calls	
Function	Where Described
sound denial check	Section 2.1.3.3.1
sound make const sound	Section 2.1.3.1.2

Table 2.1-390: sound_of_turret_drive_off Information.

2.1.3.3.25 sound_of_engine_accessory_on

This routine checks if access to the sound system is allowed, and makes the sound of the engine accessory being turned on. The priority level of Channel 5 is checked before making the sound.

Calls	
Function	Where Described
sound denial check	Section 2.1.3.3.1
sound make const sound	Section 2.1.3.1.2

Table 2.1-391: sound_of_engine_accessory_on Information.

2.1.3.3.26 sound_of_engine_accessory_already_on

This routine checks if access to the sound system is allowed, and makes the sound of the engine accessory already being on. The priority level of Channel 5 is checked before making the sound.

Calls	
Function	Where Described
sound denial check	Section 2.1.3.3.1
sound make const sound	Section 2.1.3.1.2

Table 2.1-392: sound_of_engine_accessory_already_on Information.

2.1.3.3.27 sound_of_engine_accessory_off

This routine checks if access to the sound system is allowed, and makes the sound of the engine accessory being turned off. The priority level of Channel 5 is checked before making the sound.

Calls	
Function	Where Described
sound denial check	Section 2.1.3.3.1
sound make const sound	Section 2.1.3.1.2

Table 2.1-393: sound_of_engine_accessory_off Information.

2.1.3.3.28 sound_of_random_sounds

This routine makes the random squeak, rattle, and bump sounds.

Calls	
Function	Where Described
roll_dice	/simnet/common/include/global/sim_macros.h
sound_make_const_sound	Section 2.1.3.1.2

Table 2.1-394: sound_of_random_sounds Information.

2.1.3.3.29 channel_2_check

This routine checks the priority level of Channel 2.

Calls	
Function	Where Described
sound of turret drive already on	Section 2.1.3.3.20
sound of turret power already on	Section 2.1.3.3.19

Table 2.1-395: channel_2_check Information.

2.1.3.3.30 channel_5_check

This routine checks the priority level of Channel 5.

Calls	
Function	Where Described
sound of engine accessory already on	Section 2.1.3.3.26

Table 2.1-396: channel_5_check Information.

2.1.3.4 kato_sound.c

(./simnet/release/src/vehicle/kato/kato_sound.c [kato_sound.c])

Includes:

```
"stdio.h"
"ctype.h"
"math.h"
"sim_dfns.h"
"sim_types.h"
"sim_macros.h"
"basic.h"
"obj_type.h"
"mass_stdh.h"
"dgi_stdg.h"
"kato_mem_dfn.h"
"kato_snd_dfn.h"
"sim_cig_if.h"
"fifo_dfn.h"
"fifo.h"
"libsound.h"
"libsound_dfn.h"
"libmatrix.h"
"libfail.h"
"libidc_dfn.h"
"timers_dfn.h"
"timers.h"
"kato_sound.h"
```

Defines:

```
S_MAX_TRACKS_SPEED_SOUND
S_MAX_ENGINE_SPEED_SOUND
MAX_STEALTH_SPEED_SOUND
MAX_STEALTH_VEL
MAX_TANK_VEL
MAX_APC_VEL
MAX_ENGINE_SPEED
TICKS_BETWEEN_RANDOM_SOUNDS
MAX_NUMBER_OF_RANDOM_SOUNDS
CAT_KILL_DELAY
SOUND_PORT_NUM
```

Declarations:

```
veh_sound_array      -- The listing of all possible sounds the vehicle can
                      make. Each sound is listed as an index into the
                      array.

engine_start_index
engine_vary_index
engine_stop_index
actuator_start_index
actuator_vary_index
actuator_stop_index
vel_pct
track_speed          -- Sound system speed
engine_speed         -- Last engine speed
stealth_speed
counter              -- A counter for timeouts on random sounds
sound_index          -- Pointer into the sound array
track_speed_factor   -- Fraction of max track speed
dont_use_sound        -- Set to zero if sounds are to be used
sound_of_random_sounds()
sound_denial_check()
sound_array          -- The array of random sounds
```

2.1.3.4.1 sound_denial_check

This routine returns TRUE if the sound system is not used. Access to the sound system is denied if the *dont_use_sound* flag is equal to 1, if the vehicle was destroyed, or if the *sim_state_sounds_denied()* returns FALSE. The *dont_use_sound* flag is set from the command line.

Return Values		
Return Value	Type	Meaning
TRUE	int	The use of sounds is denied
FALSE	int	The use of sounds is not denied
Calls		
Function	Where Described	
fail_death_status	Section 2.5.4.10.1	
sim_state_sounds_denied	Section 2.5.1.1.12	

Table 2.1-397: sound_denial_check Information.

2.1.3.4.2 sound_make_veh_spec_sound

This routine takes a sound request, *sound_str*, from the sound library and requests the sound from the sound system. The routine checks if access to the sound system is allowed, and queues the sound request to the fifo.

Parameters		
Parameter	Type	Where Typedef Declared
sound_str	array of char	Standard
str_len	int	Standard
Calls		
Function	Where Described	
sound_denial_check	Section 2.1.3.4.1	
fifo_enqueue	Section 2.6.8.2.1	

Table 2.1-398: sound_make_veh_spec_sound Information.

2.1.3.4.3 sound_force_veh_spec_sound

This routine takes a sound request, *sound_str*, from the sound library and requests the sound from the sound system. The routine checks if access to the sound system is allowed, and queues the sound request to the fifo. Access to the sound system may only be disallowed at the command line.

Parameters		
Parameter	Type	Where Typedef Declared
sound_str	array of char	Standard
str len	int	Standard
Calls		
Function	Where Described	
sound denial check	Section 2.1.3.4.1	
fifo enqueue	Section 2.6.8.2.1	

Table 2.1-399: sound_force_veh_spec_sound Information.

2.1.3.4.4 sound_init

This routine checks if access to the sound system is allowed, initializes the fifo, and resets the sound system.

Calls	
Function	Where Described
fifo init	Section 2.6.8.3.1
sound reset	Section 2.1.3.3.7

Table 2.1-400: sound_init Information.

2.1.3.4.5 sound_dont_use

This routine sets the *dont_use_sound* flag to 1 and prints the statement "Sound is turned off".

2.1.3.4.6 sound_simul

This routine runs the sound simulation. It checks if access to the sound system is allowed, queues sounds to the fifo, and calls for random sounds to be made.

Calls	
Function	Where Described
fifo enqueue	Section 2.6.8.2.1
sound of random sounds	Section 2.1.3.3.28

Table 2.1-401: sound_simul Information.

2.1.3.4.7 sound_reset

This routine checks if access to the sound system is allowed and resets the sound system.

Calls	
Function	Where Described
fifo_enqueue	Section 2.6.8.2.1

Table 2.1-402: sound_reset Information.

2.1.3.4.8 sound_we_just_died

This routine resets the sound system. At a later tick, the sound of catastrophic kill is requested.

Calls	
Function	Where Described
sound_reset	Section 2.1.3.3.7
timers_delay_proc	Section 2.6.3.4.1

Table 2.1-403: sound_we_just_died Information.

2.1.3.4.9 sound_of_vehicle

This routine makes the sound of the vehicle to which the Stealth vehicle is attached.

Parameters		
Parameter	Type	Where Typedef Declared
guise	ObjectType	/simnet/common/include/protocol/p_sim.h
eng_speed	REAL	/simnet/common/include/global/sim_types.h
velocity	VECTOR	/simnet/common/include/global/sim_types.h
Calls		
Function	Where Described	
sound_stop_cont_sound	Section 2.1.3.1.10	
sound_make_cont_sound	Section 2.1.3.1.2	
vec_dot_prod	Section 2.6.2.54.1	

Table 2.1-404: sound_of_vehicle Information.

2.1.3.4.12 sound_of_random_sounds

This routine makes the random sounds.

Calls	
Function	Where Described
roll dice	/simnet/common/include/global/sim_macros.h
sound make const sound	Section 2.1.3.1.2

Table 2.1-405: sound_of_random_sounds Information.

2.1.4 Controls Interface Software

Controls are defined as the operator interface to the simulation. The controls are composed of analog and digital inputs and outputs (toggle switches, steering wheels, lights, meters, etc.). This CSC contains the routines which interface with these devices. The structure of this CSC is displayed in Figure 2.1-5. This CSC consists of two third level CSCs, Controls Using IDC Boards and High Performance Analog Interface.

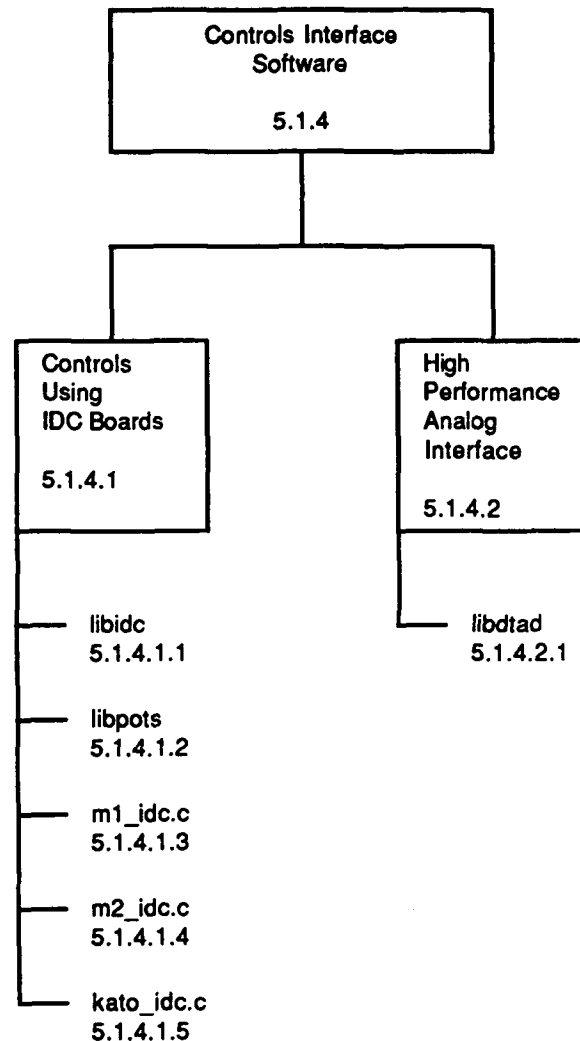


Figure 2.1-5 Controls Interface Software structure

2.1.4.1 Controls using IDC Boards

An asynchronous interface was designed to link the analog and digital inputs and outputs to the simulation host. The simulation communicates over RS232 serial lines which connect to IDC boards which provide an interface to the control panels. The software interface between analog/digital inputs and the simulation is through a shared memory segment; therefore, when an input device changes position, data in a corresponding shared memory location changes value. The simulation drives outputs (indicators, meters, lights) by

writing to the fifo utility described in section 2.6.8. Several routines (libidc, m1_idc.c, m2_idc.c, kato_idc.c) exist to deal directly with the shared memory segment. These routines initialize the memory locations and provide access to the data contained in shared memory. The library libpots was created to convert the hexadecimal number that appears in the shared memory segment to a useable value. All the potentiometers used during the simulation are calibrated off-line and their limits are saved in a file. Libpots takes the limit values and the current pot value, and returns real numbers in the format that the simulation can use (e.i., a REAL between 0 and 1 or between -1 and 1, depending on the nature of the control device). Five CSUs are associated with this CSC:

- libidc
- libpots
- m1_idc.c
- m2_idc.c
- kato_idc.c

2.1.4.1.1 libidc (./simnet/release/src/libsrc/libidc [libidc])

This library contains routines which deal directly with the shared memory segment. These routines initialize the memory locations and provide access to the data contained in shared memory.

2.1.4.1.1.1 choose_fifo.c (./simnet/release/src/libsrc/libidc /choose_fifo.c)

This file contains one procedure, `idc_choose_fifo`, which determines which FIFO to use.

Includes:

```
"stdio.h"
"sim_dfns.h"
"fifo_dfn.h"
"libidc_dfn.h"
"libidc.h"
"libmem_dfn.h"
```

2.1.4.1.1.1.1 idc_choose_fifo

This routine determines which FIFO to use, given the index to the `IDC_ARRAY` (*id*). A pointer to the appropriate FIFO segment is returned.

Parameters		
Parameter	Type	Where Typedef Declared
<code>id</code>	int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
<code>idc_ptr</code>	pointer to IDC_ENTRY	<code>idc_dfn.h</code>
<code>i</code>	int	Standard
<code>mask</code>	int	Standard
Return Values		
Return Value	Type	Meaning
<code>idc_ptr->fifo</code>	FIFO	pointer to the FIFO to be used
<code>(FIFO *) FIFO_ERROR</code>	FIFO	null pointer
Calls		
Function	Where Described	
<code>idc_get_num_idcs</code>	Section 2.1.4.1.3.1 Section 2.1.4.1.4.1 Section 2.1.4.1.5.1	

Table 2.1-406: `idc_choose_fifo` Information.

2.1.4.1.1.2 i_error.c**(./simnet/release/src/libsrc/libidc/i_error.c)**

This file contains one procedure, **libidc_error_report**, which prints an error report.

Includes:

"stdio.c"

"idc.h"

"idc_loc.h"

2.1.4.1.1.2.1 libidc_error_report

This procedure prints an error report.

Parameters		
Parameter	Type	Where Typedef Declared
func	pointer to char	Standard
sarg1	char	Standard
narg2	int	Standard

Table 2.1-407: libidc_error_report Information.

2.1.4.1.1.3 i_getact.c (./simnet/release/src/libsrc/libidc/i_getact.c)

This file contains one function, `libidc_get_action`, which retrurns the IDC actions.

Includes:

```
"stdio.h"
"idc.h"
"idc_loc.h"
```

2.1.4.1.1.3.1 libidc_get_action

This routine translates the strings in the parameter file to the appropriate number to be sent to the actions file. These values are based on IDC input or output type. The actions file is used for table look-up during the simulation. It is unique for each vehicle that is simulated.

Parameters		
Parameter	Type	Where Typedef Declared
type	pointer to char	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
ret	int	Standard
i	int	Standard
Return Values		
Return Value	Type	Meaning
ACTION_LEVEL	int	input from analog switch
ACTION_EDGE	int	input from pushbutton
ACTION_BINARY	int	input from toggle switch
ACTION_METER	int	output to meter
ACTION_DIGITAL	int	output to digital display
ACTION_SAD	int	output to situation awareness display
ACTION_SPECIAL	int	output to special device
ACTION_METER16	int	output to 16 bit meter
ACTION_MTR16HI	int	output to top 8 bits of 16 bit meter
ACTION_MTR16LO	int	output to bottom 8 bits of 16 bit meter
ACTION_LEVEL16	int	input from 16 bit analog switch
ACTION_NONE	int	unrecognized input/output

Table 2.1-408: libidc_get_action Information.

2.1.4.1.1.4 **i_getacts.c** (./simnet/release/src/libsrc/libidc/i_getacts.c)

This file contains one procedure, **idc_get_actions**, which returns an IDC action.

Includes:

```
"stdio.h"
"idc.h"
"idc_loc.h"
```

2.1.4.1.1.4.1 **idc_get_actions**

This routine returns the action for an IDC panel specified by *i*. This is a utility routine and is not used during the simulation. The return value is a character rather than the integer returned by **libidc_get_acts**.

Parameters		
Parameter	Type	Where Typedef Declared
<i>i</i>	int	Standard
Return Values		
Return Value	Type	Meaning
libidc_idc_actions[i]	char	IDC action for a panel

Table 2.1-409: **idc_get_actions** Information.

2.1.4.1.1.5 **i_getdevice.c** (./simnet/release/src/libsrc/libidc/i_getdevice.c)

This file contains one procedure, **idc_get_device_type**, which returns the type of IDC device that is being used.

Includes:

```
"stdio.h"
"idc.h"
"idc_loc.h"
```

2.1.4.1.1.5.1 **idc_get_device_type**

This routine returns a code which designates the IDC device being used. This is a utility routine and is not used during the simulation.

Return Values		
Return Value	Type	Meaning
libidc_idc_device	pointer to char	the IDC device type

Table 2.1-410: **idc_get_device_type** Information.

2.1.4.1.1.6 i_getnames.c (./simnet/release/src/libsrc/libidc/i_getnames.c)

This file contains one procedure, `idc_get_names`, which returns the *i*th element of the I/O name array.

Includes:

```
"stdio.h"  
"idc.h"  
"idc_loc.h"
```

2.1.4.1.1.6.1 idc_get_names

This routine returns a pointer to the string name in the I/O name array. The input parameter *i* is an index into the IDC table. This routine is a utility routine and is not used during the simulation.

Parameters		
Parameter	Type	Where Typedef Declared
i	int	Standard
Return Values		
Return Value	Type	Meaning
libidc_idc_names[i]	pointer to char	pointer to the string name

Table 2.1-411: `idc_get_names` Information.

2.1.4.1.1.7 `i_getoffset.c` (./simnet/release/src/libsrc/libidc/i_getoffset.c)

This file contains one procedure, `idc_get_offset`, which returns the offset in shared memory.

Includes:

```
"stdio.h"
"idc.h"
"idc_loc.h"
```

2.1.4.1.1.7.1 `idc_get_offset`

This routine returns the offset in shared memory of the current station. This routine is a utility routine and is not used during the simulation.

Return Values		
Return Value	Type	Meaning
<code>libidc_idc_offset</code>	int	IDC offset in shared memory of current station

Table 2.1-412: `idc_get_offset` Information.

2.1.4.1.1.8 `i_getport.c` (./simnet/release/src/libsrc/libidc/i_getport.c)

This file contains one procedure, `idc_get_port_name`, which returns a pointer to the name of the IDC port.

Includes:

```
"stdio.h"
"idc.h"
"idc_loc.h"
```

2.1.4.1.1.8.1 `idc_get_port_name`

This routine returns a pointer to the IDC port name. This routine is a utility routine and is not used during the simulation.

Return Values		
Return Value	Type	Meaning
<code>libidc_idc_port</code>	pointer to char	pointer to the IDC port name

Table 2.1-413: `idc_get_port_name` Information.

2.1.4.1.1.9 `i_getstat.c` (./simnet/release/src/libsrc/libidc/i_getstat.c)

This file contains one routine, `idc_get_station_description`, which returns the IDC station description.

Includes:

```
"stdio.h"
"idc.h"
"idc_loc.h"
```

2.1.4.1.1.9.1 `idc_get_station_description`

This routine returns the IDC station description. This routine is a utility routine and is not used during the simulation.

Return Values		
Return Value	Type	Meaning
<code>libidc_idc_station</code>	char	IDC station description

Table 2.1-414: `idc_get_station_description` Information.

2.1.4.1.1.10 `i_loc.c` (./simnet/release/src/libsrc/libidc/i_loc.c)

The following arrays and variables are declared:

```
libidc_idc_ibuf[80]
libidc_idc_actions[MAX_IDC_ID]
*libidc_idc_names[MAX_IDC_ID]
libidc_idc_station[80]
libidc_idc_port[80]
libidc_idc_offset
libidc_idc_device[80]
```

Includes:

```
"stdio.h"
"idc.h"
```

2.1.4.1.1.11 i_open_port.c

(./simnet/release/src/libsrc/libidc /i_open_port.c)

This file contains one procedure, **idc_open_port**, which opens an IDC port.

Includes:

```
"stdio.h"
"signal.h" Masscomp only
"sys/ioctl.h" Masscomp only
"termio.h" Masscomp only
"idc.h"
"sim_dfns.h"
sim_types.h
"fifo_dfn.h"
"fifo.h"
"idc_loc.h"
```

The procedure **idc_open_port()** is not implemented in the Butterfly. It is implemented in the Masscomp.

2.1.4.1.1.11.1 idc_open_port

This routine opens an IDC port. The file descriptor is allocated. The port is then set up for IDC operation. Port status is obtained, the old baud rate is cleared and the new baud rate is set at 4800 baud. Raw mode is set, echo is turned off, and characters are set to read. Updated status is then set.

Internal Variables		
Internal Variable	Type	Where Typedef Declared
port	int	Standard
pname	pointer to char	Standard
tio	pointer to termio	
Return Values		
Return Value	Type	Meaning
port	int	port file descriptor
Calls		
Function	Where Described	
idc_get_port_name	Section 2.1.4.1.1.8.1	

Table 2.1-415: idc_open_port Information.

2.1.4.1.1.12 i_perror.c

(./simnet/release/src/libsrc/libidc /i_perror.c)

This file contains an error reporting routine.

Includes:

"stdio.h"
"idc.h"
"idc_loc"

2.1.4.1.1.12.1 libidc_perror_report

This routine is an error reporting routine.

Parameters		
Parameters	Type	Where Typedef Declared
func	pointer to char	Standard
sarg1	pointer to char	Standard
narg2	int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
line	[80]	Standard

Table 2.1-416: libidc_perror_report Information.

2.1.4.1.1.13 i_port_stk.c

(./simnet/release/src/libsrc/libidc /i_port_stk.c)

This file contains a routine which prints an error message when ports are stuck.

Includes:

"stdio.h"
"signal.h" Masscomp only
"sys/ioctl.h" Masscomp only
"termio.h" Masscomp only
"idc.h"
"sim_dfns.h"
"sim_types.h"
"fifo_dfn.h"
"fifo.h"

2.1.4.1.1.14.1 port_stuck

This procedure prints error messages when a port is stuck. `system("stty -raw echo")` and `idc_get_port_name()` are called. This routine is not implemented in the Butterfly.

2.1.4.1.1.15 `i_raw_16_set.c` (`/simnet/release/src/libsrc/libidc/i_raw_16_set.c`)

This file sends a two byte message to a FIFO port.

Includes:

```
"stdio.h"
"signal.h" Masscomp only
"sys/ioctl.h" Masscomp only
"termio.h" Masscomp only
"idc.h"
"libidc_dfn.h"
"sim_dfns.h"
"sim_types.h"
"fifo_dfn.h"
"fifo.h"
```

`idc_set_buf[8]` is declared as a static character.

2.1.4.1.1.15.1 `idc_raw_16_set_cmd`

This routine sends a two byte message to a FIFO port, given the index to the IDC table.

The parameters are defined as follows:

```
fifo      - the FIFO port to be accessed.
id        - index into IDC table.
val       - two byte message to be sent.
```

This routine is a utilities routine and is not used during the simulation.

Parameters		
Parameter	Type	Where Typedef Declared
<i>fifo</i>	pointer to FIFO	<i>fifo_dfn.h</i>
<i>id</i>	int	Standard
<i>val</i>	short int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
<i>ret</i>	int	Standard
<i>i</i>	int	Standard
Calls		
Function	Where Described	
<i>fifo_enqueue</i>	Section 2.6.8.2.1	

Table 2.1-417: `idc_raw_16_set_cmd` Information.

2.1.4.1.1.16 i_raw_16_st2.c

(./simnet/release/src/libsrc/libidc/i_raw_16_st2.c)

This file sends a four byte message to a FIFO port.

Includes:

```
"stdio.h"
"signal.h" Masscomp only
"sys/ioctl.h" Masscomp only
"termio.h" Masscomp only
"idc.h"
"libidc_dfn.h"
"sim_dfns.h"
"sim_types.h"
"fifo_dfn.h"
"fifo.h"
```

idc_set2_buf[8] is declared as a static character.

2.1.4.1.1.16.1 idc_raw_16_set2_cmd

This routine sends a four byte message to the specified FIFO port, given an index into the IDC table. The parameters are defined as follows:

```
fifo      - the FIFO port to be accessed.
id        - index into IDC table.
val1      - first two bytes of the four byte message to be sent.
val2      - last two bytes of the four byte message to be sent.
```

This routine is a utilities routine and is not used during the simulation.

Parameters		
Parameter	Type	Where Typedef Declared
fifo	pointer to FIFO	fifo_dfn.h
id	int	Standard
val1	short int	Standard
val2	short int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
ret	int	Standard
i	int	Standard
Calls		
Function	Where Described	
fifo_enqueue	Section 2.6.8.2.1	

Table 2.1-418: idc_raw_16_set2_cmd Information.

2.1.4.1.1.17 i_raw_set.c

(./simnet/release/src/libsrc/libidc /i_raw_set.c)

This file sends a one byte message to a FIFO port.

Includes:

```
"stdio.h"
"signal.h" Masscomp only
"sys/ioctl.h" Masscomp only
"termio.h" Masscomp only
"idc.h"
"libidc_dfn.h"
"sim_dfns.h"
"sim_types.h"
"fifo_dfn.h"
"fifo.h"
```

idc_set_buf[8] is declared as a static character.

2.1.4.1.1.17.1 idc_raw_set_cmd

This routine sends a one byte message to the specified FIFO port, given an index into the IDC table. The parameters are defined as follows:

<i>fifo</i>	- the FIFO port to be accessed.
<i>id</i>	- index into IDC table, representing a particular IDC panel.
<i>val</i>	- one byte message to be sent.

This routine is a utilities routine and is not used during the simulation.

Parameters		
Parameter	Type	Where Typedef Declared
<i>fifo</i>	pointer to FIFO	fifo_dfn.h
<i>id</i>	int	Standard
<i>val</i>	short int	Standard
Calls		
Function	Where Described	
<i>fifo_enqueue</i>	Section 2.6.8.2.1	

Table 2.1-419: idc_raw_set_cmd Information.

2.1.4.1.1.18 i_readbody.c

(./simnet/release/src/libsrc/libidc /i_readbody.c)

This file contains a routine which reads the IDC parameter file body.

Includes:

```
"stdio.h"
"idc.h"
"idc_loc.h"
```

2.1.4.1.1.18.1 libidc_read_idc_parameter_body

This routine reads the IDC parameter file body. *name* is a pointer to the file name. This routine is a utility routine and is not used during the simulation.

Parameters		
Parameter	Type	Where Typedef Declared
fp	pointer to FILE	Standard
name	pointer to char	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
id	int	Standard
type[80]	char	Standard
idname[80]	char	Standard
action	int	Standard
libidc strsave	pointer to char	Standard
Calls		
Function	Where Described	
libidc strsave	Section 2.1.4.1.1.8.8	
libidc error report	Section 2.1.4.1.1.8.8	
libidc get actions	Section 2.1.4.1.1.8.8	

Table 2.1-420: libidc_read_idc_parameter_body Information.

2.1.4.1.1.19 i_readfile.c

(./simnet/release/src/libsrc/libidc/i_readfile.c)

This file contains a routine which reads an IDC parameter file.

Includes:

```
"stdio.h"
"idc.h"
"idc_loc.h"
```

DEBUG is defined as FALSE.

2.1.4.1.1.19.1 read_idc_parameter_file

This routine reads the IDC parameter file. *name* is a pointer to the name of the parameter file. The parameter file maps logical names to shared memory offsets. This routine is a utility routine and is not used during the simulation.

Parameters		
Parameter	Type	Where Typedef Declared
name	pointer to char	Standard
Internal Variables		
Internal Variables	Type	Where Typedef Declared
fp	pointer to FILE	Standard
i	int	Standard
Calls		
Function	Where Described	
libidc perror report	Section 2.1.4.1.1.12.1	
libidc_read_idc_parameter_header	Section 2.1.4.1.1.20.1	
libidc_read_idc_parameter_body	Section 2.1.4.1.1.18.1	

Table 2.1-421: read_idc_parameter_file Information.

2.1.4.1.1.20 i_readhead.c

(./simnet/release/src/libsrc/libidc/i_readhead.c)

This file contains a routine which reads the IDC parameter file header.

Includes:

"stdio.h"
"idc.h"
"idc_loc.h"

DEBUG is defined as FALSE.

2.1.4.1.1.20.1 libidc_read_idc_parameter_header

This routine reads the IDC parameter file header. *name* is a pointer to the file name. This routine is a utility routine and is not used during the simulation.

Parameters		
Parameter	Type	Where Typedef Declared
fp	pointer to FILE	Standard
name	pointer to char	Standard
Calls		
Function	Where Described	
libidc_error_report	Section 2.1.4.1.1.2.1	

Table 2.1-422: libidc_read_idc_parameter_header Information.

2.1.4.1.1.21 i_reset.c

(./simnet/release/src/libsrc/libidc/i_reset.c)

This file contains a routine which resets the FIFO port.

Includes:

```
"stdio.h"
"signal.h" Masscomp only
"sys/ioctl.h" Masscomp only
"termio.h" Masscomp only
"idc.h"
"libidc_dfn.h"
"sim_dfns.h"
"sim_types.h"
"fifo_dfn.h"
"fifo.h"
```

idc_reset_buf[8] is declared as a static character.

2.1.4.1.1.21.1 idc_reset_cmd

This routine resets the FIFO buffer designated by *fifo*. The reset command is interpreted by the IDC hardware.

Parameters		
Parameter	Type	Where Typedef Declared
<i>fifo</i>	pointer to FIFO	<i>fifo_dfn.h</i>
Calls		
Function	Where Described	
<i>fifo_enqueue</i>	Section 2.6.8.2.1	

Table 2.1-423: idc_reset_cmd Information.

2.1.4.1.1.22 i_strsave.c

(./simnet/release/src/libsrc/libidc /i_strsave.c)

This file contains a routine which saves a string.

Includes:

"stdio.h"
 "idc.h"
 "idc_loc.h"

2.1.4.1.1.22.1 libidc_strsave

A string is copied and saved. This routine then returns a pointer to that string. This routine is a utility routine and is not used during the simulation.

Parameters		
Parameter	Type	Where Typedef Declared
f	pointer to char	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
t	pointer to char	Standard
Return Values		
Return Value	Type	Meaning
t	pointer to char	pointer to a string of interest
Calls		
Function	Where Described	
libidc_perror_report	Section 2.1.4.1.1.12.1	

Table 2.1-424: libidc_strsave Information.

2.1.4.1.1.23 idc_loc.h

(./simnet/release/src/libsrc/libidc /idc_loc.h)

This file defines constants and declares external variables and procedures which are used within libidc.

2.1.4.1.1.24 `init.c`**(`/simnet/release/src/libsrc/libidc /init.c`)**

This file contains routines which initialize the IDCs at the start of a simulation.

Includes:

```
"stdio.h"
"font1.h"
"sim_dfns.h"
"fifo_dfn.h"
"fifo.h"
"libidc_dfn.h"
"libidc.h"
"libmem_dfn.h"
```

The following external variables are declared:

```
reset_idc_msg [IDC_RESET_SIZE]
set_idc_msg [IDC_SET_SIZE]
set_16_idc_msg [IDC_SET_16_SIZE]
set2_16_idc_msg [IDC_SET2_16_SIZE]
```

2.1.4.1.1.24.1 `idc_init`

This routine initializes the IDC shared memory. The IDC_ARRAY of IDC_ENTRIES is initialized. The FIFO segment of shared memory is initialized. The IDC ports are reset. Vehicle-specific initializations are accomplished via `idc_vch_spec_init()` so that the appropriate lights and meters are set for startup.

Calls	
Function	Where Described
<code>idc_array_init</code>	Section 2.1.4.1.3.2 Section 2.1.4.1.4.2 Section 2.1.4.1.5.2
<code>idc_fifo_init</code>	Section 2.1.4.1.1.24.2
<code>idc_reset</code>	Section 2.1.4.1.1.24.4
<code>idc_vch_spec_init</code>	Section 2.1.4.1.3.4 Section 2.1.4.1.4.4 Section 2.1.4.1.5.4

Table 2.1-425: `idc_init` Information.

2.1.4.1.1.24.2 idc_fifo_init

This routine searches through the IDC_ENTRIES and initializes the FIFO portion of the shared memory.

Internal Variables		
Internal Variable	Type	Where Typedef Declared
idc_ptr	pointer to IDC_ENTRY	libidc_dfn.h
i	int	Standard
Calls		
Function	Where Described	
idc_get_num_idcs	Section 2.1.3.1.3.1 Section 2.1.3.1.4.1 Section 2.1.3.1.5.1	
fifo_init	Section 2.6.8.5.1	

Table 2.1-426: idc_fifo_init Information.

2.1.4.1.1.24.3 idc_fifo_uninit

This routine uninitializes the FIFO portion of the shared memory.

Internal Variables		
Internal Variable	Type	Where Typedef Declared
idc_ptr	pointer to IDC_ENTRY	libidc_dfn.h
i	int	Standard
Calls		
Function	Where Described	
idc_get_num_idcs	Section 2.1.3.1.3.1 Section 2.1.3.1.4.1 Section 2.1.3.1.5.1	
fifo_uninit	Section 2.6.8.3.2	

Table 2.1-427: idc_fifo_uninit Information.

2.1.4.1.1.24.4 idc_reset

This routine resets the IDC inputs and outputs.

Calls	
Function	Where Described
idc_reset_input	Section 2.1.4.1.1.24.5
idc_reset_output	Section 2.1.4.1.1.24.6

Table 2.1-428: idc_reset Information.

2.1.4.1.1.24.5 idc_reset_input

This routine clears the IDC portion of the shared memory.

Internal Variables		
Internal Variable	Type	Where Typedef Declared
i	int	Standard
Calls		
Function	Where Described	
idc_get_idc_share_size	Section 2.1.4.1	

Table 2.1-429: idc_reset_input Information.

2.1.4.1.1.24.6 idc_reset_output

This routine sends a reset command to the IDCs.

Internal Variables		
Internal Variable	Type	Where Typedef Declared
idc_ptr	pointer to IDC_ENTRY	libidc_dfn.h
i	int	Standard
Calls		
Function	Where Described	
idc_get_num_idcs	Section 2.1.3.1.3.1 Section 2.1.3.1.4.1 Section 2.1.3.1.5.1	
fifo_uninit	Section 2.6.8.3.2	

Table 2.1-430: idc_reset_output Information.

2.1.4.1.1.25 op_16_set.c

(./simnet/release/src/libsrc/libidc/op_16_set.c)

This file sends a two byte message to output.

Includes:

```
"stdio"
"sim_dfns.h"
"fifo_dfn.h"
"fifo.h:
"libidc_dfn.h"
"libidc.h"
"libmem_dfn.h"
```

2.1.4.1.1.25.1 idc_output_16_set

This routine sends a two byte message to a serial port and copies the data to shared memory. The parameters are defined as follows:

id - an index into the IDC table.

val - the data to be sent.

Parameters		
Parameter	Type	Where Typedef Declared
<i>id</i>	int	Standard
<i>val</i>	short int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
<i>fifo</i>	pointer to FIFO	fifo_dfn.h
Calls		
Function	Where Described	
<i>idc_choose_fifo</i>	Section 2.1.4.1.1.1	
<i>fifo_enqueue</i>	Section 2.6.8.2.1	

Table 2.1-431: idc_output_16_set Information.

2.1.4.1.1.26 op_16_set2.c

(./simnet/release/src/libsrc/libidc/op_18_set2.c)

This file sends a four byte message to output.

Includes:

```
"stdio"
"sim_dfns.h"
"fifo_dfn.h"
"fifo.h"
"libidc_dfn.h"
"libidc.h"
"libmem_dfn.h"
```

2.1.4.1.1.26.1 idc_output_16_set2

This routine sends a four byte message to a serial port and copies the data to shared memory. The parameters are defined as follows:

id - an index into the IDC table.
val1 - the first two bytes of the data to be sent.
val2 - the last two bytes of the data to be sent.

Parameters		
Parameter	Type	Where Typedef Declared
<i>id</i>	int	Standard
<i>val1</i>	short int	Standard
<i>val2</i>	short int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
<i>fifo</i>	pointer to FIFO	fifo_dfn.h
Calls		
Function	Where Described	
<i>idc_choose_fifo</i>	Section 2.1.4.1.1.1.1	
<i>fifo_enqueue</i>	Section 2.6.8.2.1	

Table 2.1-432: idc_output_16_set2 Information.

2.1.4.1.1.27 op_rest.c

(./simnet/release/src/libsrc/libidc /op_rest.c)

This file contains one procedure, **idc_output_restore**.

Includes:

```
"stdio.h"
"sim_dfns.h"
"fifo_dfn.h"
"fifo.h"
"libidc_dfn.h"
"libidc.h"
"libmem_dfn.h"
```

2.1.4.1.1.27.1 idc_output_restore

This routine reads the value in shared memory and sends it to the serial port. *id* designates the location in shared memory that is of interest.

Parameters		
Parameter	Type	Where Typedef Declared
id	int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
fifo	pointer to FIFO	fifo_dfn.h
Calls		
Function	Where Described	
fifo_enqueue	Section 2.6.8.2.1	

Table 2.1-433: **idc_output_restore** Information.

2.1.4.1.1.28 op_rest.c.c

(./simnet/release/src/libsrc/libidc/op_rest.c.c)

This file contains one procedure, **idc_output_restore_cond**.

Includes:

```
"stdio.h"
"sim_dfns.h"
"fifo_dfn.h"
"fifo.h"
"libidc_dfn.h"
"libidc.h"
"libmem_dfn.h"
```

2.1.4.1.1.28.1 idc_output_restore_cond

This routine conditionally calls the output restore routine. The parameters are defined as follows:

id - designates the location in shared memory.
cond - specifies the condition under which the output restore routine should be called.

Parameters		
Parameter	Type	Where Typedef Declared
<i>id</i>	int	Standard
<i>val</i>	char	Standard
Calls		
Function	Where Described	
<i>idc_output_restore</i>	Section 2.1.4.1.1.27.1	

Table 2.1-434: **idc_output_restore_cond** Information.

2.1.4.1.1.29 op_set.c

(./simnet/release/src/libsrc/libidc /op_set.c)

Includes:

"stdio.h"
 "sim_dfns.h"
 "fifo_dfn.h"
 "fifo.h"
 "libidc_dfn.h"
 "libidc.h"
 "libmem_dfn.h"

2.1.4.1.1.29.1 idc_output_set_ns_cond

This routine determines which FIFO port to use, stores the data as well as the address of the data in shared memory, and queues the data to the appropriate FIFO port. The parameters are defined as follows:

id - the address in shared memory.
 val - the data to be sent.

Parameters		
Parameter	Type	Where Typedef Declared
id	int	Standard
val	char	Standard
Internal Variables		
Internal Variables	Type	Where Typedef Declared
fifo	pointer to FIFO	fifo_dfn.h
Calls		
Function	Where Described	
idc_choose_fifo	Section 2.1.4.1.1.1.1	
fifo_enqueue	Section 2.6.8.2.1	

Table 2.1-435: idc_output_set_ns_cond Information.

2.1.4.1.1.30 op_set_c.c

(./simnet/release/src/libsrc/libidc /op_set_c.c)

This file contains one procedure, **idc_output_set_cond**, which sets output under certain conditions.

Includes:

```
"stdio.h"
"sim_dfns.h"
"fifo_dfn.h"
"fifo.h"
"libidc_dfn.h"
"libidc.h"
"libmem_dfn.h"
```

2.1.4.1.1.30.1 idc_output_set_cond

This routine sets output under certain conditions. The parameters are defined as follows:

cond - condition under which to set outputs.
id - index into IDC array.
val - the data to be sent.

Parameters		
Parameter	Type	Where Typedef Declared
<i>cond</i>	int	Standard
<i>id</i>	int	Standard
<i>val</i>	char	Standard
Calls		
Function	Where Described	
<i>idc_output_set</i>	Section 2.1.4	

Table 2.1-436: **idc_output_set_cond** Information.

2.1.4.1.1.31 op_set_ns.c

(./simnet/release/src/libsrc/libidc /op_set_ns.c)

This file contains one procedure, **idc_output_set_ns_cond**, which sends an output request to an appropriate IDC with no update of shared memory.

Includes:

```
"stdio.h"
"sim_dfns.h"
"fifo_dfn.h"
"fifo.h"
"libidc_dfn.h"
"libidc.h"
"libmem_dfn.h"
```

2.1.4.1.1.31.1 idc_output_set_ns_cond

This routine sends an output request to an appropriate IDC with no update of shared memory. The parameters are defined as follows:

id - index into IDC array.
val - the data to be sent.

Parameters		
Parameter	Type	Where Typedef Declared
<i>id</i>	int	Standard
<i>val</i>	char	Standard
Internal Variables		
Internal Variables	Type	Where Typedef Declared
<i>fifo</i>	pointer to FIFO	fifo_dfn.h
Calls		
Function	Where Described	
<i>fifo_enqueue</i>	Section 2.6.8.2.1	

Table 2.1-437: **idc_output_set_ns_cond** Information.

2.1.4.1.1.32 op_set_ns_c.c

(./simnet/release/src/libsrc/libidc /op_set_ns_c.c)

This file contains one procedure, **idc_output_set_ns_cond**, which conditionally sends an output request to an appropriate IDC with no update of shared memory.

Includes:

```
"stdio.h"
"sim_dfns.h"
"fifo_dfn.h"
"fifo.h"
"libidc_dfn.h"
"libidc.h"
"libmem_dfn.h"
```

2.1.4.1.1.32.1 idc_output_set_ns_cond

This routine conditionally sends an output request to an appropriate IDC with no update of shared memory. The parameters are defined as follows:

cond - designates the condition under which to send an output request.
id - index into the IDC array.
val - the data to be sent.

Parameters		
Parameter	Type	Where Typedef Declared
cond	bit	Standard
id	int	Standard
val	char	Standard
Calls		
Function	Where Described	
idc_output_set_ns	Section 2.1.4.1.31.1	

Table 2.1-438: idc_output_set_ns_cond Information.

2.1.4.1.1.33 respond.c

(./simnet/release/src/libsrc/libidc /respond.c)

This file contains a routine which echoes a value in shared memory.

Includes:

```
"stdio.h"
"sim_dfns.h"
"sim_macros.h"
"fifo_dfn.h"
"fifo.h"
"libidc_dfn.h"
"libidc.h"
"libmem_dfn.h"
```

2.1.4.1.1.33.1 idc_respond

This routine looks at an address in shared memory designated by *id*. If a value is present, it will be echoed. This routine is a utility routine and is not used during the simulation.

Parameters		
Parameter	Type	Where Typedef Declared
id	int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
val	char	Standard

Table 2.1-439: idc_respond Information.

2.1.4.1.2 libpots

(./simnet/release/src/libsrc/libpots [libpots])

Potentiometers are used during the simulation to transmit analog signals from the controls. The potentiometer returns an analog voltage (between 0 - ff hex) which is converted to a digital value. The simulation requires these values to be translated to real numbers between -1.0 and 1.0 (or 0 and 1.0). The potentiometers also must be calibrated so that their full range maps to the appropriate values. This functionality is provided by this CSU.

2.1.4.1.2.1 p_clamp.c

(./simnet/release/src/libsrc/libpots/p_clamp.c)

This file contains a routine which crops a value.

Includes:

```
"stdio.h"
"sim_types.h"
"sim_dfns.h"
"sim_macros.h"
"libpots.h"
```

2.1.4.1.2.1.1 pots_clamp_pot_between

This routine crops the input value *pot*. If *pot* is below the lower limit, the lower limit is returned. If it is above the upper limit, the upper limit is returned. *val1* and *val2* define the required range. If *pot* falls within the range, it is returned.

Parameters		
Parameter	Type	Where Typedef Declared
pot	int	Standard
val1	int	Standard
val2	int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
mini	int	Standard
maxi	int	Standard
Return Values		
Return Value	Type	Meaning
pot	int	new value of the potentiometer

Table 2.1-440: pots_clamp_pot_between Information.

2.1.4.1.2.2 p_lcr.c

(./simnet/release/src/libsrc/libpots/p_lcr.c)

This file contains a routine which clamps a pot value to the left, center, or right, depending on its value.

Includes:

```
"stdio.h"
"sim_types.h"
"sim_dfns.h"
"sim_macros.h"
"libpots.h"
```

2.1.4.1.2.2.1 pots_scale_lcr

The value *pot* is first clamped to the left or right. A dead zone of size *tolerance* is created in each direction from the center position (*center*). The appropriate real value is then assigned between -1.0 and 1.0.

Parameters		
Parameter	Type	Where Typedef Declared
pot	int	Standard
left	int	Standard
center	int	Standard
right	int	Standard
tolerance	int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
new left	int	Standard
new right	int	Standard
retval	REAL	sim_types.h
Return Values		
Return Value	Type	Meaning
retval	REAL	calibrated pot value
Calls		
Function	Where Described	
pots clamp pots between	Section 2.1.4.1.2.1.1	

Table 2.1-441: pots_scale_lcr Information.

2.1.4.1.2.3 p_lr_both.c

(./simnet/release/src/libsrc/libpots/p_lr_both.c)

This file contains a routine which scales a potentiometer value to between -1.0 and 1.0.

Includes:

```
"stdio.h"
"sim_types.h"
"sim_dfns.h"
"sim_macros.h"
"libpots.h"
```

2.1.4.1.2.3.1 pots_scale_lr_both

This routine scales the potentiometer value to between -1.0 and 1.0. *pot* is the potentiometer value, and *left* and *right* define the range.

Parameters		
Parameter	Type	Where Typedef Declared
pot	int	Standard
left	int	Standard
right	int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
retval	REAL	sim_types.h
Return Values		
Return Value	Type	Meaning
retval	REAL	calibrated value of potentiometer value
Calls		
Function	Where Described	
pots clamp pot between	Section 2.1.4.1.2.1.1	

Table 2.1-442: pots_scale_lr_both Information.

2.1.4.1.2.4 p_lr_pos.c

(./simnet/release/src/libsrc/libpots/p_lr_pos.c)

This file contains a routine which scales a potentiometer value to between 0.0 and 1.0.

Includes:

```
"stdio.h"
"sim_types.h"
"sim_dfns.h"
"sim_macros.h"
"libpots.h"
```

2.1.4.1.2.4.1 pots_scale_lr_pos

This routine scales the potentiometer value to between 0.0 and 1.0. *pot* is the potentiometer value, and *left* and *right* define the range.

Parameters		
Parameter	Type	Where Typedef Declared
pot	int	Standard
left	int	Standard
right	int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
retval	REAL	sim_types.h
Return Values		
Return Value	Type	Meaning
retval	REAL	calibrated value
Calls		
Function	Where Described	
pots_clamp_pot_between	Section 2.1.4.1.2.1.1	

Table 2.1-443: pots_scale_lr_pos Information.

2.1.4.1.2.5 p_three.c

(./simnet/release/src/libsrc/libpots/p_three.c)

This file contains a routine which verifies that three potentiometer values are valid.

Includes:

```
"stdio.h"
"sim_types.h"
"sim_dfns.h"
"sim_macros.h"
"libpots.h"
```

2.1.4.1.2.5.1 pots_check_three

This routine verifies that three potentiometer values are valid. It determines if the middle value is in fact the middle value. If it is, the routine returns TRUE. If it is not, it returns FALSE and prints the line number (*line*) in the calibration file where this error occurred.

Parameters		
Parameter	Type	Where Typedef Declared
line	int	Standard
first	int	Standard
second	int	Standard
third	int	Standard
Return Values		
Return Value	Type	Meaning
TRUE	int	valid pot values
FALSE	int	invalid pot values

Table 2.1-444: pots_check_three Information.

2.1.4.1.2.6 p_two.c

(./simnet/release/src/libsrc/libpots/p_two.c)

This file contains a routine which determines if two potentiometer values are valid.

Includes:

```
"stdio.h"
"sim_types.h"
"sim_dfns.h"
"sim_macros.h"
"libpots.h"
```

2.1.4.1.2.6.1 pots_check_two

This routine determines if two potentiometer values are valid. If the two values (*first* and *second*) are not equal, the routine returns TRUE. If the two values are equal, the routine returns FALSE and prints the line number (*line*) in the calibration file where the error occurred.

Parameters		
Parameter	Type	Where Typedef Declared
line	int	Standard
first	int	Standard
second	int	Standard

Table 2.1-445: pots_check_two Information.

2.1.4.1.2.7 libpots.h

(./simnet/release/src/libsrc/libpots/libpots.h)

This file declares the following procedures for use outside of libpots.

```
pots_clamp_pot_between()
pots_scale_lcr()
pots_scale_lr_both()
pots_scale_lr_pos()
pots_check_three()
pots_check_two()
```

2.1.4.1.3 m1_idc.c

(./simnet/release/src/vehicle/m1/src/m1_idc.c [m1_idc.c])

This CSU contains routines which initialize the IDCs for the M1 simulator.

Includes:

```
"stdio.h"
"sim_dfns.h"
"fifo_dfn.h"
"fifo.h"
"libidc.h"
"libidc_dfn.h"
"libmem.h"
"libmem_dfn.h"
"m1_mem_dfn.h"
"m1_driv_pn.h"
"m1_turr_pn.h"
"m1_idc.h"
```

Defined:

NUM_IDCS

Declared:

idc_array[NUM_IDCS]

2.1.4.1.3.1 idc_get_num_idcs

This routine returns the number of IDCs that are used.

Return Values		
Return Value	Type	Meaning
NUM_IDCS	int	the number of IDCs used

Table 2.1-446: idc_get_num_idcs Information.

2.1.4.1.3.2 idc_array_init

This routine initializes the IDC array.

2.1.4.1.3.3 idc_invert_outputs

This routine sets the outputs.

Calls	
Function	Where Described
idc_output_set	Section 2.1.4

Table 2.1-447: idc_invert_outputs Information.

2.1.4.1.3.4 idc_veh_spec_init

This routine is stubbed out but contains no functionality.

2.1.4.1.4 m2_idc.c

(./simnet/release/src/vehicle/m2/src/m2_idc.c [m2_idc.c])

This CSU contains the routines which initialize the IDCs for the M2 simulator.

Includes:

```
"stdio.h"
"sim_dfns.h"
"fifo_dfn.h"
"fifo.h"
"libidc.h"
"libidc_dfn.h"
"libmem.h"
"libmem_dfn.h"
"m2_mem_dfn.h"
"m2_driv_pn.h"
"m2_turr_pn.h"
"m2_idc.h"
```

Defined:

NUM_IDCS

Declared:

```
idc_array[NUM_IDCS]
reticle_init_val
```

2.1.4.1.4.1 idc_get_num_idcs

This routine returns the number of IDCs that are used.

Return Values		
Return Value	Type	Meaning
NUM_IDCS	int	the number of IDC's used

Table 2.1-448: idc_get_num_idcs Information.

2.1.4.1.4.2 idc_array_init

This routine initializes the IDC array.

2.1.4.1.4.3 idc_invert_outputs

This routine sets the outputs.

Calls	
Function	Where Described
idc_output_set	Section 2.1.4

Table 2.1-449: idc_invert_outputs Information.

2.1.4.1.3.4 `idc_veh_spec_init`

This routine initializes the reticles associated with the M2 simulator.

Calls	
Function	Where Described
<code>idc_output_set</code>	Section 2.1.4

Table 2.1-450: `idc_veh_spec_init` Information.

2.1.4.1.3.5 `idc_set_reticle_init_val`

This routine sets the initial values for the reticles. The values are set to *num*.

Parameters		
Parameter	Type	Where Typedef Declared
<code>num</code>	<code>int</code>	Standard

Table 2.1-451: `idc_set_reticle_init_val` Information.

2.1.4.1.5 kato_idc.c

(./simnet/release/src/vehicle/kato/src/kato_idc.c [kato_idc.c])

This CSU contains the routines which initialize the IDCs for the Stealth.

Includes:

```
"stdio.h"
"sim_dfns.h"
"fifo_dfn.h"
"fifo.h"
"libidc.h"
"libidc_dfn.h"
"libmem.h"
"libmem_dfn.h"
"kato_mem_dfn.h"
"kato_hard.h" (for use with spaceball)
"kato_soft.h"
```

Defined:

NUM_IDCS

Declared:

idc_array[NUM_IDCS]

2.1.4.1.5.1 idc_get_num_idcs

This routine returns the number of IDCs which are used.

Return Values		
Return Value	Type	Meaning
NUM_IDCS	int	the number of IDC's used

Table 2.1-452: idc_get_num_idcs Information.

2.1.4.1.5.2 idc_array_init

This routine initializes the IDC array.

2.1.4.1.5.3 idc_veh_spec_init

This routine is stubbed out but contains no functionality.

2.1.4.2 High Performance Analog Interface

The High Performance Analog Interface CSC implements the interface to the high performance analog interface. This additional input device was required because the 8 bit analog to digital converter provided on the IDC boards did not have sufficient resolution for some of the potentiometers on the M1 simulation. The only CSU required is libdtad.

2.1.4.2.1 libdtad

(./simnet/release/src/libsrc/libdtad [libdtad])

Libdtad provides support for the data translation card.

2.1.4.2.1.1 ain.c

(./simnet/release/src/libsrc/libdtad/ain.c)

Includes "dtad_loc.h"

2.1.4.2.1.1.1 ain

This routine returns the analog input value given the channel number. *channel* represents the analog input channel (0-7) on the Data Translation card. These channels are mapped to potentiometers on the card. If access to the card is not enabled, this routine returns a value of 0 (the lowest value of the input range). If the polling loop times out, -1 is returned.

Note that since the AD card's busy bit is not asserted quickly enough before polling starts, a delay is necessary to prevent reading the AD before its conversion has started.

Parameters		
Parameter	Type	Where Typedef Declared
channel	int	Standard
Internal Variables		
Variable	Type	Where Typedef Declared
i	int	Standard
temp	int	Standard
Return Values		
Return Value	Type	Meaning
0	int	lowest value of analog input; access is not enabled
-1	int	the card is bad
Dtad_read()	int	analog input value of 0-4095; linearly related to the analog input voltage
Calls		
Function	Where Described	
Dtad_start	dtad.h	
Dtad_pull	dtad.h	
Dtad_read	dtad.h	

Table 2.1-453: ain Information.

2.1.4.2.1.2 attatch.c

(./simnet/release/src/libsrc/libdtad/attatch.c)

Includes:

"dtad_loc.h"

2.1.4.2.1.2.1 dtad_attach

This routine attaches the Dtad device to the process. The hardware is mapped into the host's address space so it can be accessed. The routine returns 0 if it is successful and -1 if it fails.

Internal Variables		
Variable	Type	Where Typedef Declared
sbrk()	pointer to char (MASSCOMP only)	Standard
Return Values		
Return Value	Type	Meaning
0	int	procedure successful
-1	int	procedure failed
Calls		
Function	Where Described	
map_mbregion (Butterfly Machine only)	Standard Butterfly function	
pmapm (Masscomp Machine only)	Standard Masscomp function	

Table 2.1-454: dtad_attach Information.

2.1.4.2.1.3 cur_minus12.c

(.simnet/release/src/libsrc/libdtad/cur_minus12.c)

Includes:

"dtad_loc.h"

2.1.4.2.1.3.1 current_minus12

This routine reads the -12 volt power supply and returns the current voltage (which maps to the value of a specified potentiometer). The value is rounded to the nearest tenth of a volt because of limited measurement accuracy.

Internal Variables		
Variable	Type	Where Typedef Declared
voltage	register int	Standard
Return Values		
Return Value	Type	Meaning
voltage/10.0	float	-12 volt power supply voltage
Calls		
Function	Where Described	
ain	Section 2.1.4.2.1.1.1	

Table 2.1-455: current_minus12 Information.

2.1.4.2.1.4 cur_plus12.c

(.simnet/release/src/libsrc/libdtad/cur_plus12.c)

Includes:

"dtad_loc.h"

2.1.4.2.1.4.1 current_plus12

This routine reads the +12 volt power supply and returns the current voltage (which maps to the value of a specified potentiometer). The value is rounded to the nearest tenth of a volt because of limited measurement accuracy.

Internal Variables		
Variable	Type	Where Typedef Declared
voltage	register int	Standard
Return Values		
Return Value	Type	Meaning
voltage/10.0	float	+12 volt power supply voltage
Calls		
Function	Where Described	
ain	Section 2.1.4.2.1.1.1	

Table 2.1-456: current_plus12 Information.

2.1.4.2.1.5 cur_plus5.c (./simnet/release/src/libsrc/libdtad/cur_plus5.c)

Includes:

"dtad_loc.h"

2.1.4.2.1.5.1 current_plus5

This routine reads the +5 volt power supply and returns the current voltage (which maps to the value of a specified potentiometer). The value is rounded to the nearest hundredth of a volt because of limited measurement accuracy.

Internal Variables		
Variable	Type	Where Typedef Declared
voltage	register int	Standard
Returns		
Return Value	Type	Meaning
voltage/100.0	float	+5 volt power supply voltage
Calls		
Function	Where Described	
ain	Section 2.1.4.2.1.1.1	

Table 2.1-457: current_plus5 Information.

2.1.4.2.1.6 cur_temp.c (./simnet/release/src/libsrc/libdtad/cur_temp.c)

Includes:
"dtad_loc.h"

2.1.4.2.1.6.1 current_temperature

This routine returns the current temperature in degrees C. The temperature from the thermistor is related to the voltage. The value is rounded to the nearest whole number because of limited measurement accuracy. The returned value is obtained from a lookup table.

rt is the calculated thermistor resistance.
temp is the temperature-related voltage.
ref is the reference-related voltage.

Internal Variables		
Variable	Type	Where Typedef Declared
i	register int	Standard
rt	register int	Standard
temp	register int	Standard
ref	register int	Standard
Return Values		
Return Value	Type	Meaning
temperature_table[i][0]	float	
Calls		
Function	Where Described	
ain	Section 2.1.4.2.1.1.1	

Table 2.1-458: current_temperature Information.

2.1.4.2.1.7 data.c (./simnet/release/src/libsrc/libdtad/data.c)

Include "dtad_loc.h"

Declarations:
mb_mapregion (for SIMBFLY machine only)
dtad_env (for MASSCOMP machine only)
*dtadvaddr (for MASSCOMP machine only)
*pdtad
really_use_the_dtad

This file defines the temperature table used by the **current_temperature()** routine.

2.1.4.2.1.8 detatch.c

(/simnet/release/src/libsrc/libdtad/detatch.c)

Includes "dtad_loc.h"

2.1.4.2.1.8.1 dtad_detach

This routine detaches (or unmaps) the Dtad device from the process.

Calls	
Function	Where Described
unmap_mregion (SIMBFLY machine only)	Standard Butterfly function
punmap (MASSCOMP machine only)	Standard Masscomp function

Table 2.1-459: dtad_detach Information.

2.1.4.2.1.9 dtad_loc.h

(/simnet/release/src/libsrc/libdtad/dtad_loc.h)

Includes:

"stdio.h"
 "multibus.h" (Butterfly Machine only)
 "mbregion.h" (Butterfly Machine only)
 "signal.h" (Masscomp Machine only)
 "setjmp.h" (Masscomp Machine only)
 "sys/types.h" (Masscomp Machine only)
 "sim_types.h"
 "sim_dfns.h"
 "dtad.h"

Declarations:

errno
 pdtad
 really_use_the_dtad
 dtadvaddr
 temperature_table[][2]

2.1.4.2.1.10 init.c

(/simnet/release/src/libsrc/libdtad/init.c)

Includes "dtad_loc.h"

2.1.4.2.1.10.1 dtad_signal_handler

This routine is defined for the Masscomp Machine only. It is used to catch the bus error signal that is sent if the dtad is not in the system when accessed in the attach routine. It executes a longjmp (standard Unix call) to the location set up by **dtad_init()**.

2.1.4.2.1.10.2 dtad_init

This routine initializes the Dtad device and is called before attempting to use the device. The Dtad device is attached to the calling process and probed to determine whether it is present in the system. The *really use the dtad* flag is set FALSE if the Dtad is not in the system. A message is printed about whether the device was found. The routine returns -1 if there is an error, and 0 if not.

Internal Variables		
Variable	Type	Where Typedef Declared
temp	char	Standard
Return Values		
Return Value	Type	Meaning
-1	int	error
0	int	procedure successful
Calls		
Function	Where Described	
dtad_attach	Section 2.1.4.2.1.2.1	

Table 2.1-460: dtad_init Information.

2.1.4.2.1.11 unittest.c

(./simnet/release/src/libsrc/libdtad/uninit.c)

Includes "dtad_loc.h"

2.1.4.2.1.11.1 dtad_uninit

This routine uninitializes the Dtad device.

Calls	
Function	Where Described
dtad_detach	Section 2.1.4.2.1.8.1

Table 2.1-461: dtad_uninit Information.

2.1.5 Status Panel Interface Software

The structure of the Status Panel Interface Software CSC is depicted in Figure 2.1-6.

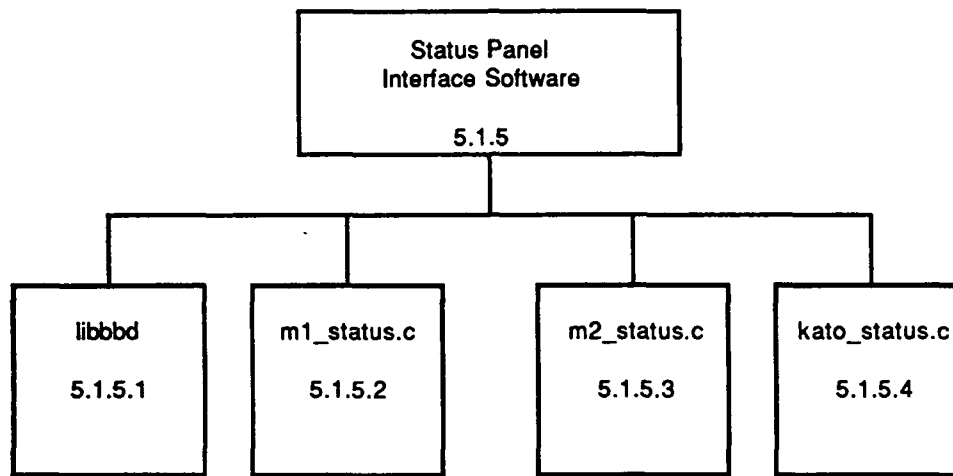


Figure 2.1-6: Structure of the Status Panel Interface Software CSC.

To aid in the successful operation and maintenance of a simulator, several components are monitored in software (e.g., cig status, sound system status, IDC card status, and network status). Failure of a component is indicated on an indicator panel associated with each simulator. The status data is obtained and maintained in the vehicle specific code in `m1_status.c`, `m2_status.c`, and `kato_status.c`. The software interface between the status data and the indicator panel is accomplished through the library `libbbd`. The hardware interface between the simulation and the status indicator panel is accomplished through the Burr Brown MP830-72 TTL I/O card. `Libbbd` provides the functions to communicate with this I/O card.

This functionality is realized by the following CSUs:

- `libbbd`
- `m1_status.c`
- `m2_status.c`
- `kato_status.c`

2.1.5.1 libbbd

(./simnet/release/src/libsrc/libbbd [libbbd])

The library `libbbd` allows the software interface between the status data and the indicator panel. It also provides the functions to communicate with the I/O card that allows the hardware interface between the simulation and the status indicator panel.

2.1.5.1.1 attach.c

(./simnet/release/src/libsrc/libbbd/attach.c)

This file contains routines which attach the bbd device to the process.

Includes:

"bbd_loc.h"

2.1.5.1.1.1 bbd_attach

This routine attaches the Burr Brown device to the process. A value of 0 is returned if the attach process is successful, and a value of -1 is returned otherwise.

If a Butterfly machine is used:

Return Values		
Return Value	Type	Meaning
-1	int	failed
0	int	successful
Calls		
Function	Where Described	
map_mbregion	Standard Butterfly function	

Table 2.1-462: bbd_attach Information for the Butterfly Machine.

If a Masscomp machine is used:

Internal Variables		
Internal Variable	Type	Where Typedef Declared
sbrk()	pointer to char	Standard
Return Values		
Return Value	Type	Meaning
-1	int	failed
0	int	successful

Table 2.1-463: bbd_attach Information for the Masscomp Machine.

2.1.5.1.2 bbd_loc.h

(./simnet/release/src/libsrc/libbbd/bbd_loc.h)

This file defines a number of constants and declares several external variables for use in libbbd.

2.1.5.1.3 bit_in.c

(./simnet/release/src/libsrc/libbbd/bit_in.c)

This file contains a procedure which reads a specified bit.

Includes:

"bbd_loc.h"

2.1.5.1.3.1 bbd_bit_in

This routine reads a bit specified by the argument *bitnum*.. This routine returns the value in the least significant bit of the return value.

Parameters		
Parameter	Type	Where Typedef Declared
bitnum	int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
val	int	Standard
pn	int	Standard
bn	int	Standard
Return Values		
Return Value	Type	Meaning
0	int	the device hasn't been initialized or the value in the bit is equal to 0
val	int	the value in the least significant bit of the return value

Table 2.1-464: bbd_bit_in Information.

2.1.5.1.4 bit_out.c

(./simnet/release/src/libsrc/libbbd/bit_out.c)

This file contains a routine which writes to the specified bit.

Includes:

bbd_loc.h"

2.1.5.1.4.1 bbd_bit_out

This routine writes to a specified bit. The arguments are the bit number of the specified bit (*bitnum*) and the value (in the least significant bit) to which to set the bit. If *val* is equal to zero, a 0 is written to the bit. If *val* has a nonzero value, a 1 is written to the bit.

Parameters		
Parameter	Type	Where Typedef Declared
bitnum	int	Standard
val	int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
pn	int	Standard
bn	int	Standard
temp	int	Standard

Table 2.1-465: bbd_bit_out Information.

2.1.5.1.5 byte_in.c

(./simnet/release/src/libsrc/libbbd/byte_in.c)

This file contains a routine which reads a specified byte.

Includes:

bbd_loc.h"

2.1.5.1.5.1 bbd_byte_in

This routine reads a byte specified by the argument *portnum.*, which ranges in value from 0 to 8. The byte from the specified port is returned.

Parameters		
Parameter	Type	Where Typedef Declared
portnum	int	Standard
Return Values		
Return Value	Type	Meaning
0	int	bbd not being used
pbbd->p[portnum] & 0xff	int	byte from desired port

Table 2.1-466: bbd_byte_in Information.

2.1.5.1.6 byte_out.c

(./simnet/release/src/libsrc/libbbd/byte_out.c)

This file contains a routine which writes to a specified byte.

2.1.5.1.6.1 bbd_byte_out

This routine writes to a byte specified by *portnum.*, which can have a value of 0 to 8. *val* is the value to output and can range from 0 to 255.

Parameters		
Parameter	Type	Where Typedef Declared
portnum	int	Standard
val	int	Standard

Table 2.1-467: bbd_byte_out Information.

2.1.5.1.7 control_in.c

(./simnet/release/src/libsrc/libbbd/control_in.c)

The control register determines if the ports are configured for inputs or outputs. This file contains a routine which reads the value of a control register to determine the status of the ports.

2.1.5.1.7.1 bbd_control_in

This routine reads the value of a control register to determine its status.

Internal Variables		
Internal Variable	Type	Where Typedef Declared
val	int	Standard
Return Values		
Return Value	Type	Meaning
0	int	device not in use or the value in the register is equal to 0
val	int	the value in the control register

Table 2.1-468: bbd_control_in Information.

2.1.5.1.8 control_out.c

(./simnet/release/src/libsrc/libbbd/control_out.c)

This file contains a routine which writes to a control register.

2.1.5.1.8.1 bbd_control_out

This routine writes to a control register. *val* specifies the bit pattern to be written to the control register. This bit pattern determines the input/output status of the ports.

Parameters		
Parameter	Type	Where Typedef Declared
val	int	Standard

Table 2.1-469: bbd_control_out Information.

2.1.5.1.9 data.c

(./simnet/release/src/libsrc/libbbd/data.c)

Several variables are declared in this file for use in libbbd.

2.1.5.1.10 detach.c

(./simnet/release/src/libsrc/libbbd//detach.c)

This file contains a routine which detaches the bbd device from the process.

2.1.5.1.10.1 bbd_detach

This routine detaches the bbd device from the process.

Calls	
Function	Where Described
unmap_mbregion (Butterfly only)	Standard Butterfly function

Table 2.1-470: bbd_detach Information.

2.1.5.1.11 init.c

(./simnet/release/src/libsrc/libbbd /init.c)

This file contains routines which initialize the bbd device for all outputs.

Includes:

"bbd_loc.h"

2.1.5.1.11.1 bbd_init

This routine initializes the bbd device. The argument is a pointer to a character array containing the initial port values. The card is initialized for all outputs. A value of 0 is returned if successful and -1 is returned otherwise. A message is printed which indicates whether or not the device was found.

Parameters		
Parameter	Type	Where Typedef Declared
p	pointer to char	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
temp	char	Standard
i	int	Standard

Table 2.1-471: bbd_init Information.

If a Butterfly machine is used:

Return Values		
Return Value	Type	Meaning
-1	int	failure
0	int	success
Calls		
Function	Where Described	
bbd_attach	Section 2.1.5.1.1.1	
bbd_control_out	Section 2.1.5.1.8.1	
bbd_byte_out	Section 2.1.5.1.6.1	

Table 2.1-472: bbd_init Information for the Butterfly Machine.

If a Masscomp machine is used:

Return Values		
Return Value	Type	Meaning
-1	int	failure
0	int	success
Calls		
Function	Where Described	
bbd_attach	Section 2.1.5.1.1.1	
bbd_control_out	Section 2.1.5.1.8.1	
bbd_byte_out	Section 2.1.5.1.6.1	

Table 2.1-473: bbd_init Information for the Masscomp Machine.

2.1.5.1.11.2 bbd_signal_handler

This routine handles the bus error signal which can result when determining if the BBD card is plugged into the system.

2.1.5.1.12 statistics.c

(./simnet/release/src/libsrc/libbbd/statistics.c)

This file is a stub.

2.1.5.1.13 status.c

(./simnet/release/src/libsrc/libbbd/status.c)

This file contains one routine, **status_out**, which writes status bits to a status panel.

Includes:

"bbd_loc.h"

2.1.5.1.13 status_out

This routine writes status bits to a status panel. The more significant status bits are at lower status addresses in the status word. *stat* is the bit pattern which is written to the Burr Brown card and is used to turn status lights on and off.

Parameters		
Parameter	Type	Where Typedef Declared
stat	unsigned	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
pi	pointer to register unsigned char	Standard
po	pointer to register unsigned char	Standard

Table 2.1-474: status_out Information.

2.1.5.1.16 unittest.c

(./simnet/release/src/libsrc/libbbd/uninit.c)

This file contains a routine which uninitializes the Burr Brown device.

2.1.5.1.16.1 bbd_uninit

This routine uninitializes the Burr Brown device and sets it to all inputs.

Internal Variables		
Internal Variable	Type	Where Typedef Declared
i	int	Standard
Calls		
Function	Where Described	
bbd_detach	Section 2.1.5.1.10.1	
bbd_rtc_statistics	Section 2.1.5.1.12	
bbd_bit_out	Section 2.1.5.1.4.1	
bbd_control_out	Section 2.1.5.1.8.1	

Table 2.1-475: bbd_uninit Information.

2.1.5.2 m1_status.c

(./simnet/release/src/vehicle/m1/src/m1_status.c [m1_status.c])

Includes:

```
"sim_types.h"
"sim_dfn.h"
"net/network.h"
"libnetwork.h"
"libdc.h"
"libdc_dfn.h"
"libmem_dfn.h"
"m1_mem_dfn.h"
"m1_turr_pn.h"
"m1_driv_pn.h"
"m1_ammo_pn.h"
"m1_soun_pn.h"
"m1_status.h"
```

unsigned variable declarations and initialization:

```
frame_counter
equipment_status = RED_HOST | RED_CIG | RED_SOUND | RED_DRIVER |
                  RED_TURRET | RED_AMMO | RED_12P | RED_12N |
                  RED_5P | RED_NET
```

int variable declarations and initialization:

```
need_to_set_host_red = FALSE
need_to_set_cig_red = FALSE
need_to_set_driver_red = FALSE
need_to_set_turret_red = FALSE
need_to_set_ammo_red = FALSE
need_to_set_sound_red = FALSE
need_to_set_voltage12P_red = FALSE
need_to_set_voltage12N_red = FALSE
need_to_set_voltage5_red = FALSE
need_to_set_net_red = FALSE
print_status_messages = FALSE      - controls status printing
dtad_failed = FALSE
```

unsigned char variable declarations and initialization:

```
health_host = 1
health_cig = 1
health_sound = 1
health_driver = 1
health_ammo = 1
health_turret = 1
health_net = 1
```

float variable declarations and initialization:

```
voltage12P = 12.0
voltage12N = -12.0
voltage5 = 5.0
temperature = 70.0
```

char array declarations and initialization:

```
st_com[] = {HEAD00, HEAD01, IDC_STATUS}
st_sound[] = {/*HEAD00,HEAD01,*/ SOUND_STATUS}
```

2.1.5.2.1 what_is_voltage12P

This routine returns the variable *voltage12P*.

Return Values		
Return Value	Type	Meaning
voltage12P	float	voltage

Table 2.1-476: what_is_voltage12P Information.

2.1.5.2.2 what_is_voltage12N

This routine returns the variable *voltage12N*.

Return Values		
Return Value	Type	Meaning
voltage12N	float	voltage

Table 2.1-477: what_is_voltage12N Information.

2.1.5.2.3 what_is_voltage5

This routine returns the variable *voltage5*.

Return Values		
Return Value	Type	Meaning
voltage5	float	voltage

Table 2.1-478: what_is_voltage5 Information.

2.1.5.2.4 what_is_temperature

This routine returns the variable *temperature*.

Return Values		
Return Value	Type	Meaning
temperature	float	temperature.

Table 2.1-479: what_is_temperature Information.

2.1.5.2.5 status_preset

This routine presets entries in the shared memory so the first execution of *status_simul* does not indicate failures.

2.1.5.2.6 status_init

This routine presets entries in the shared memory by calling status_preset, and then calls status_out.

Calls	
Function	Where Described
status_out	Section 2.1.5.1.13

Table 2.1-480: status_init Information.

2.1.5.2.7 status_simul

This routine monitors the status of the devices during the simulation.

Calls	
Function	Where Described
monitor_status	Section 2.1.5.2.23

Table 2.1-481: status_simul Information.

2.1.5.2.8 status_print_temp_and_supplies

This routine displays the current values of the temperature and the power supplies, as measured and calculated from the I/O card on the console.

Calls	
Function	Where Described
current_temperature	Section 2.1.4.2.1.6.1
current_plus5	Section 2.1.4.2.1.5.1
current_plus12	Section 2.1.4.2.1.4.1
current_minus12	Section 2.1.4.2.1.3.1

Table 2.1-482: status_print_temp_and_supplies Information.

2.1.5.2.9 driver_dead

This routine indicates that the driver IDC is dead.

Parameters		
Parameter	Type	Where Typedef Declared
p	pointer to char	Standard
Calls		
Function	Where Described	
nprintf	Section 2.1.1.3.1.34.1	

Table 2.1-483: driver_dead Information.

2.1.5.2.10 turret_dead

This routine indicates that the turret IDC is dead.

Parameters		
Parameter	Type	Where Typedef Declared
p	pointer to char	Standard
Calls		
Function	Where Described	
nprintf	Section 2.1.1.3.1.34.1	

Table 2.1-484: turret_dead Information.

2.1.5.2.11 ammo_dead

This routine indicates that the ammo IDC is dead.

Parameters		
Parameter	Type	Where Typedef Declared
p	pointer to char	Standard
Calls		
Function	Where Described	
nprintf	Section 2.1.1.3.1.34.1	

Table 2.1-485: ammo_dead Information.

2.1.5.2.12 cig_dead

This routine indicates that the CIG is dead.

Parameters		
Parameter	Type	Where Typedef Declared
p	pointer to char	Standard
Calls		
Function	Where Described	
nprintf	Section 2.1.1.3.1.34.1	

Table 2.1-486: cig_dead Information.

2.1.5.2.13 net_dead

This routine indicates a network problem.

Parameters		
Parameter	Type	Where Typedef Declared
p	pointer to char	Standard
Calls		
Function	Where Described	
nprintf	Section 2.1.1.3.1.34.1	

Table 2.1-487: net_dead Information.

2.1.5.2.14 ser_dead

This routine indicates that the ser (hpsm) card is dead.

Parameters		
Parameter	Type	Where Typedef Declared
p	pointer to char	Standard
Calls		
Function	Where Described	
nprintf	Section 2.1.1.3.1.34.1	

Table 2.1-488: ser_dead Information.

2.1.5.2.15 dtad_dead

This routine indicates that the dtad card is dead.

Parameters		
Parameter	Type	Where Typedef Declared
p	pointer to char	Standard
Calls		
Function	Where Described	
nprintf	Section 2.1.1.3.1.34.1	

Table 2.1-489: dtad_dead Information.

2.1.5.2.16 sound_dead

This routine indicates that the sound system is dead.

Parameters		
Parameter	Type	Where Typedef Declared
p	pointer to char	Standard
Calls		
Function	Where Described	
nprintf	Section 2.1.1.3.1.34.1	

Table 2.1-490: sound_dead Information.

2.1.5.2.17 plus12_dead

This routine indicates that the +12 volt supply has gone awry.

Parameters		
Parameter	Type	Where Typedef Declared
p	pointer to char	Standard
Calls		
Function	Where Described	
nprintf	Section 2.1.1.3.1.34.1	

Table 2.1-491: plus12_dead Information.

2.1.5.2.18 minus12_dead

This routine indicates that the -12 volt supply has gone awry.

Parameters		
Parameter	Type	Where Typedef Declared
p	pointer to char	Standard
Calls		
Function	Where Described	
nprintf	Section 2.1.1.3.1.34.1	

Table 2.1-492: minus12_dead Information.

2.1.5.2.19 plus5_dead

This routine indicates that the +5 volt supply has gone awry.

Parameters		
Parameter	Type	Where Typedef Declared
p	pointer to char	Standard
Calls		
Function	Where Described	
nprintf	Section 2.1.1.3.1.34.1	

Table 2.1-493: plus5_dead Information.

2.1.5.2.20 enable_status_printing

This routine enables printing of status messages by setting the variable print_status_messages to TRUE.

2.1.5.2.21 disable_status_printing

This routine disables printing of status messages by setting the variable print_status_messages to FALSE.

2.1.5.2.22 cig_failed_fsm

This routine is a finite state machine that is called in the event of CIG failure. It indicates that the CIG is dead and checks the equipment status every second, and broadcasts the equipment status onto the net every minute.

Internal Variables		
Internal Variable	Type	Where Typedef Declared
now	long	Standard
temp	int	Standard
seconds	int	Standard
i	int	Standard
Calls		
Function	Where Described	
network_can_i_really_use_network	Section 2.1.1.3.2.27.1	
net_current_time		
network_get_net_handle	Section 2.1.1.3.2.12.1	
cig_dead	Section 2.1.5.2.12	
monitor_status	Section 2.1.5.2.23	

Table 2.1-494: cig_failed_fsm Information.

2.1.5.2.23 monitor_status

This routine checks the status of the various subsystems. Each subsystem is checked on a different frame. A flag is set for each subsystem that fails. On the final frame of the cycle the flags are checked and the lights are updated.

Parameters		
Parameter	Type	Where Typedef Declared
which	int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
last heartbeat	int	Standard
new heartbeat	int	Standard
Calls		
Function	Where Described	
driver dead	Section 2.1.5.2.9	
fifo enqueue	Section 2.6.8.2.1	
turret dead	Section 2.1.5.2.10	
ammo dead	Section 2.1.5.2.11	
network_can_i_really_use_network	Section 2.1.1.3.2.27.1	
net heartbeat		
network get net handle	Section 2.1.1.3.2.12.1	
net dead	Section 2.1.5.2.13	
ser heartbeat	Section 2.6.7.2.1	
ser dead	Section 2.1.5.2.14	
net xmt failed	Section 2.1.1.3.1.24.1	
set xmt failed	Section 2.1.1.3.1.56.1	
dtad dead	Section 2.1.5.2.15	
sound dead	Section 2.1.5.2.16	
current temperature	Section 2.1.4.2.1.6.1	
current plus12	Section 2.1.4.2.1.4.1	
plus12 dead	Section 2.1.5.2.17	
current minus12	Section 2.1.4.2.1.3.1	
minus12 dead	Section 2.1.5.2.18	
current plus5	Section 2.1.4.2.1.5.1	
plus5 dead	Section 2.1.5.2.19	
status out	Section 2.1.5.1.13	

Table 2.1-495: monitor_status Information.

2.1.5.3 m2_status.c

(./simnet/release/src/vehicle/m2/src/m2_status.c [m2_status.c])

Includes:

```
"sim_types.h"
"sim_dfns.h"
"net/network.h"
"libnetwork.h"
"libidc.h"
"libidc_dfn.h"
"libmem_dfn.h"
"m2_mem_dfn.h"
"m2_turr_pn.h"
"m2_driv_pn.h"
"m2_soun_pn.h"
"m2_status.h"
"fifo.h"
"fifo_dfn.h"
"status.h"
"ser_status.h"
"dtad.h"
```

unsigned variable declarations and initialization:

```
frame_counter
equipment_status = RED_HOST | RED_CIG | RED_SOUND | RED_DRIVER |
                  RED_TURRET | RED_12P | RED_12N |
                  RED_5P | RED_NET
```

int variable declarations and initialization:

```
need_to_set_host_red = FALSE
need_to_set_cig_red = FALSE
need_to_set_driver_red = FALSE
need_to_set_turret_red = FALSE
need_to_set_sound_red = FALSE
need_to_set_voltage12P_red = FALSE
need_to_set_voltage12N_red = FALSE
need_to_set_voltage5_red = FALSE
need_to_set_net_red = FALSE
print_status_messages = FALSE      - controls status printing
dtad_failed = FALSE
```

unsigned char variable declarations and initialization:

```
health_host = 1
health_cig = 1
health_sound = 1
health_driver = 1
health_turret = 1
health_net = 1
```

float variable declarations and initialization:

```
voltage12P = 12.0
voltage12N = -12.0
voltage5 = 5.0
temperature = 70.0
```

char array declarations and initialization:

```
st_com[] = {HEAD00, HEAD01, IDC_STATUS}
st_sound[] = {/*HEAD00, HEAD01, */ SOUND_STATUS}
```

extern int variable declarations:

```
netxmt_failed
```

2.1.5.3.1 what_is_voltage12P

This routine returns the variable *voltage12P*.

Return Values		
Return Value	Type	Meaning
voltage12P	float	voltage

Table 2.1-496: what_is_voltage12P Information.

2.1.5.3.2 what_is_voltage12N

This routine returns the variable *voltage12N*.

Return Values		
Return Value	Type	Meaning
voltage12N	float	voltage

Table 2.1-497: what_is_voltage12N Information.

2.1.5.3.3 what_is_voltage5

This routine returns the variable *voltage5*.

Return Values		
Return Value	Type	Meaning
voltage5	float	voltage

Table 2.1-498: what_is_voltage5 Information.

2.1.5.3.4 what_is_temperature

This routine returns the variable *temperature*.

Return Values		
Return Value	Type	Meaning
temperature	float	temperature.

Table 2.1-499: what_is_temperature Information.

2.1.5.3.5 status_preset

This routine presets entries in the shared memory so the first execution of status_simul does not indicate failures.

2.1.5.3.6 status_init

This routine presets entries in the shared memory by calling status_preset, and then calls status_out.

Calls	
Function	Where Described
status_out	Section 2.1.5.1.13

Table 2.1-500: status_init Information.

2.1.5.3.7 status_simul

This routine monitors the status of devices during the simulation.

Calls	
Function	Where Described
monitor_status	Section 2.1.5.3.23

Table 2.1-501: status_simul Information.

2.1.5.3.8 status_print_temp_and_supplies

This routine displays the current values of the temperature and the power supplies, as measured and calculated from the I/O card on the console.

Calls	
Function	Where Described
current temperature	Section 2.1.4.2.1.6.1
current plus5	Section 2.1.4.2.1.5.1
current plus12	Section 2.1.4.2.1.4.1
current minus12	Section 2.1.4.2.1.3.1

Table 2.1-502: status_print_temp_and_supplies Information.

2.1.5.3.9 driver_dead

This routine indicates that the driver IDC is dead.

Parameters		
Parameter	Type	Where Typedef Declared
p	pointer to char	Standard
Calls		
Function	Where Described	
nprintf	Section 2.1.1.3.1.34.1	

Table 2.1-503: driver_dead Information.

2.1.5.3.10 turret_dead

This routine indicates that the turret IDC is dead.

Parameters		
Parameter	Type	Where Typedef Declared
p	pointer to char	Standard
Calls		
Function	Where Described	
nprintf	Section 2.1.1.3.1.34.1	

Table 2.1-504: turret_dead Information.

2.1.5.3.11 cig_dead

This routine indicates that the CIG is dead.

Parameters		
Parameter	Type	Where Typedef Declared
p	pointer to char	Standard
Calls		
Function	Where Described	
nprintf	Section 2.1.1.3.1.34.1	

Table 2.1-505: cig_dead Information.

2.1.5.3.12 net_dead

This routine indicates a network problem.

Parameters		
Parameter	Type	Where Typedef Declared
p	pointer to char	Standard
Calls		
Function	Where Described	
nprintf	Section 2.1.1.3.1.34.1	

Table 2.1-506: net_dead Information.

2.1.5.3.13 ser_dead

This routine indicates that the ser (hpsm) card is dead.

Parameters		
Parameter	Type	Where Typedef Declared
p	pointer to char	Standard
Calls		
Function	Where Described	
nprintf	Section 2.1.1.3.1.34.1	

Table 2.1-507: ser_dead Information.

2.1.5.3.14 dtad_dead

This routine indicates that the dtad card is dead.

Parameters		
Parameter	Type	Where Typedef Declared
p	pointer to char	Standard
Calls		
Function	Where Described	
nprintf	Section 2.1.1.3.1.34.1	

Table 2.1-508: dtad_dead Information.

2.1.5.3.15 sound_dead

This routine indicates that the sound system is dead.

Parameters		
Parameter	Type	Where Typedef Declared
p	pointer to char	Standard
Calls		
Function	Where Described	
nprintf	Section 2.1.1.3.1.34.1	

Table 2.1-509: sound_dead Information.

2.1.5.3.16 plus12_dead

This routine indicates that the +12 volt supply has gone awry.

Parameters		
Parameter	Type	Where Typedef Declared
p	pointer to char	Standard
Calls		
Function	Where Described	
nprintf	Section 2.1.1.3.1.34.1	

Table 2.1-510: plus12_dead Information.

2.1.5.3.17 minus12_dead

This routine indicates that the -12 volt supply has gone awry.

Parameters		
Parameter	Type	Where Typedef Declared
p	pointer to char	Standard
Calls		
Function	Where Described	
nprintf	Section 2.1.1.3.1.34.1	

Table 2.1-511: minus12_dead Information.

2.1.5.3.18 plus5_dead

This routine indicates that the +5 volt supply has gone awry.

Parameters		
Parameter	Type	Where Typedef Declared
p	pointer to char	Standard
Calls		
Function	Where Described	
nprintf	Section 2.1.1.3.1.34.1	

Table 2.1-512: plus5_dead Information.

2.1.5.3.19 enable_status_printing

This routine enables printing of status messages by setting the variable print_status_messages to TRUE.

2.1.5.3.20 disable_status_printing

This routine disables printing of status messages by setting the variable print_status_messages to FALSE.

2.1.5.3.21 cig_failed_fsm

This routine is a finite state machine that is called in the event of CIG failure. It indicates that the CIG is dead and checks the equipment status every second, and broadcasts the equipment status onto the net every minute.

Internal Variables		
Internal Variable	Type	Where Typedef Declared
now	long	Standard
temp	int	Standard
seconds	int	Standard
i	int	Standard
Calls		
Function	Where Described	
network_can_i_really_use_network	Section 2.1.1.3.2.27.1	
net current time		
network_get_net_handle	Section 2.1.1.3.2.12.1	
cig_dead	Section 2.1.5.3.12	
monitor status	Section 2.1.5.3.23	

Table 2.1-513: cig_failed_fsm Information.

2.1.5.3.22 monitor_status

This routine checks the status of the various subsystems. Each subsystem is checked on a different frame. A flag is set for each subsystem that fails. On the final frame of the cycle the flags are checked and the lights are updated.

Parameters		
Parameter	Type	Where Typedef Declared
which	int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
last_heartbeat	int	Standard
new_heartbeat	int	Standard
Calls		
Function	Where Described	
driver_dead	Section 2.1.5.3.9	
fifo_enqueue	Section 2.6.8.2.1	
turret_dead	Section 2.1.5.3.10	
network_can_i_really_use_network	Section 2.1.1.3.2.27.1	
net_heartbeat		
network_get_net_handle	Section 2.1.1.3.2.12.1	
net_dead	Section 2.1.5.3.13	
ser_heartbeat	Section 2.6.7.2.1	
ser_dead	Section 2.1.5.3.14	
net_xmt_failed	Section 2.1.1.3.1.24.1	
set_xmt_failed	Section 2.1.1.3.1.56.1	
dtad_dead	Section 2.1.5.3.15	
sound_dead	Section 2.1.5.3.16	
current_temperature	Section 2.1.4.2.1.6.1	
current_plus12	Section 2.1.4.2.1.4.1	
plus12_dead	Section 2.1.5.3.17	
current_minus12	Section 2.1.4.2.1.3.1	
minus12_dead	Section 2.1.5.3.18	
current_plus5	Section 2.1.4.2.1.5.1	
plus5_dead	Section 2.1.5.3.19	
status_out	Section 2.1.5.1.13	

Table 2.1-514: monitor_status Information.

2.1.5.4 kato_status.c

(./simnet/Release/src/vehicle/kato/src/kato_status.c [kato_status.c])

Includes:

```
"sim_types.h"
"sim_dfns.h"
"net/network.h"
"libnetwork.h"
"libidc.h"
"libidc_dfn.h"
"libmem_dfn.h"
"kato_mem_dfn.h"
"kato_hard.h"          if USE_SPACEBALL not defined
"kato_soft.h"
"kato_soun_pn.h"
"kato_status.h"
"fifo.h"
"fifo_dfn.h"
"status.h"
"cmc.h"
"cmc_status.h"
"ser_status.h"
"dtad.h"
```

unsigned variable declarations and initialization:

```
frame_counter
equipment_status = RED_HOST | RED_CIG | RED_SOUND | RED_HARD |
                  RED_SOFT | RED_12P | RED_12N | RED_5P | RED_NET
```

int variable declarations and initialization:

```
need_to_set_host_red = FALSE
need_to_set_cig_red = FALSE
need_to_set_hard_red = FALSE
need_to_set_soft_red = FALSE
need_to_set_sound_red = FALSE
need_to_set_voltage12P_red = FALSE
need_to_set_voltage12N_red = FALSE
need_to_set_voltage5_red = FALSE
need_to_set_net_red = FALSE
print_status_messages = FALSE          - controls status printing
dtad_failed = FALSE
```

unsigned char variable declarations and initialization:

```
health_host = 1
health_cig = 1
health_sound = 1
health_hard = 1
health_soft = 1
health_net = 1
```

float variable declarations and initialization:

```
voltage12P = 12.0  
voltage12N = -12.0  
voltage5 = 5.0  
temperature = 70.0
```

char array declarations and initialization:

```
st_com[] = {HEAD00, HEAD01, IDC_STATUS}  
st_sound[] = {/*HEAD00, HEAD01, */ SOUND_STATUS}
```

extern int variable declarations:

```
netxmt_failed
```

2.1.5.4.1 what_is_voltage12P

This routine returns the variable *voltage12P*.

Return Values		
Return Value	Type	Meaning
voltage12P	float	voltage.

Table 2.1-515: what_is_voltage12P Information.

2.1.5.4.2 what_is_voltage12N

This routine returns the variable *voltage12N*.

Return Values		
Return Value	Type	Meaning
voltage12N	float	voltage.

Table 2.1-516: what_is_voltage12N Information.

2.1.5.4.3 what_is_voltage5

This routine returns the variable *voltage5*.

Return Values		
Return Value	Type	Meaning
voltage5	float	voltage.

Table 2.1-517: what_is_voltage5 Information.

2.1.5.4.4 what_is_temperature

This routine returns the variable *temperature*.

Return Values		
Return Value	Type	Meaning
temperature	float	temperature.

Table 2.1-518: what_is_temperature Information.

2.1.5.4.5 status_preset

This routine presets entries in the shared memory so the first execution of status_simul does not indicate failures.

2.1.5.4.6 status_init

This routine presets entries in the shared memory by calling status_preset, and then calls status_out.

Calls	
Function	Where Described
status_out	Section 2.1.5.1.13

Table 2.1-519: status_init Information.

2.1.5.4.7 status_simul

This routine monitors devices during the simulation.

Calls	
Function	Where Described
monitor_status	Section 2.1.5.4.23

Table 2.1-520: status_simul Information.

2.1.5.4.8 status_print_temp_and_supplies

This routine displays the current values of the temperature and the power supplies, as measured and calculated from the I/O card on the console.

Calls	
Function	Where Described
current_temperature	Section 2.1.4.2.1.6.1
current_plus5	Section 2.1.4.2.1.5.1
current_plus12	Section 2.1.4.2.1.4.1
current_minus12	Section 2.1.4.2.1.3.1

Table 2.1-521: status_print_temp_and_supplies Information.

2.1.5.4.9 hard_dead

This routine indicates that the hard IDC is dead. This routine is compiled only if `USE_SPACEBALL` is not defined.

Parameters		
Parameter	Type	Where Typedef Declared
p	pointer to char	Standard
Calls		
Function	Where Described	
nprintf	Section 2.1.1.3.1.34.1	

Table 2.1-522: hard_dead Information.

2.1.5.4.10 soft_dead

This routine indicates that the soft IDC is dead.

Parameters		
Parameter	Type	Where Typedef Declared
p	pointer to char	Standard
Calls		
Function	Where Described	
nprintf	Section 2.1.1.3.1.34.1	

Table 2.1-523: soft_dead Information.

2.1.5.4.11 cig_dead

This routine indicates that the CIG is dead.

Parameters		
Parameter	Type	Where Typedef Declared
p	pointer to char	Standard
Calls		
Function	Where Described	
nprintf	Section 2.1.1.3.1.34.1	

Table 2.1-524: cig_dead Information.

2.1.5.4.12 net_dead

This routine indicates a network problem.

Parameters		
Parameter	Type	Where Typedef Declared
p	pointer to char	Standard
Calls		
Function	Where Described	
nprintf	Section 2.1.1.3.1.34.1	

Table 2.1-525: net_dead Information.

2.1.5.4.13 ser_dead

This routine indicates that the ser (hpsm) card is dead.

Parameters		
Parameter	Type	Where Typedef Declared
p	pointer to char	Standard
Calls		
Function	Where Described	
nprintf	Section 2.1.1.3.1.34.1	

Table 2.1-526: ser_dead Information.

2.1.5.4.14 dtad_dead

This routine indicates that the dtad card is dead.

Parameters		
Parameter	Type	Where Typedef Declared
p	pointer to char	Standard
Calls		
Function	Where Described	
nprintf	Section 2.1.1.3.1.34.1	

Table 2.1-527: dtad_dead Information.

2.1.5.4.15 sound_dead

This routine indicates that the sound system is dead.

Parameters		
Parameter	Type	Where Typedef Declared
p	pointer to char	Standard
Calls		
Function	Where Described	
nprintf	Section 2.1.1.3.1.34.1	

Table 2.1-528: sound_dead Information.

2.1.5.4.16 plus12_dead

This routine indicates that the +12 volt supply has gone awry.

Parameters		
Parameter	Type	Where Typedef Declared
p	pointer to char	Standard
Calls		
Function	Where Described	
nprintf	Section 2.1.1.3.1.34.1	

Table 2.1-529: plus12_dead Information.

2.1.5.4.17 minus12_dead

This routine indicates that the -12 volt supply has gone awry.

Parameters		
Parameter	Type	Where Typedef Declared
p	pointer to char	Standard
Calls		
Function	Where Described	
nprintf	Section 2.1.1.3.1.34.1	

Table 2.1-530: minus12_dead Information.

2.1.5.4.18 plus5_dead

This routine indicates that the +5 volt supply has gone awry.

Parameters		
Parameter	Type	Where Typedef Declared
p	pointer to char	Standard
Calls		
Function	Where Described	
nprintf	Section 2.1.1.3.1.34.1	

Table 2.1-531: plus5_dead Information.

2.1.5.4.19 enable_status_printing

This routine enables printing of status messages by setting the variable `print_status_messages` to TRUE.

2.1.5.4.20 disable_status_printing

This routine disables printing of status messages by setting the variable `print_status_messages` to FALSE.

2.1.5.4.21 cig_failed_fsm

This routine is a finite state machine that is called in the event of CIG failure. It indicates that the CIG is dead and checks the equipment status every second, and broadcasts the equipment status onto the net every minute.

Internal Variables		
Internal Variable	Type	Where Typedef Declared
now	long	Standard
temp	int	Standard
seconds	int	Standard
i	int	Standard
Calls		
Function	Where Described	
net current time		
network get net handle	Section 2.1.1.3.2.12.1	
cig dead	Section 2.1.5.4.12	
monitor status	Section 2.1.5.4.23	

Table 2.1-532: cig_failed_fsm Information.

2.1.5.4.22 monitor_status

This routine checks the status of the various subsystems. Each subsystem is checked on a different frame. A flag is set for each subsystem that fails. On the final frame of the cycle the flags are checked and the lights are updated.

Parameters		
Parameter	Type	Where Typedef Declared
which	int	Standard
Internal Variables		
Internal Variable	Type	Where Typedef Declared
last heartbeat	int	Standard
new heartbeat	int	Standard
Calls		
Function	Where Described	
hard dead	Section 2.1.5.4.9	
fifo enqueue	Section 2.6.8.2.1	
soft dead	Section 2.1.5.4.10	
cmc heartbeat		
net dead	Section 2.1.5.4.13	
ser heartbeat	Section 2.6.7.2.1	
ser dead	Section 2.1.5.4.14	
net xmt failed	Section 2.1.1.3.1.24.1	
set xmt failed	Section 2.1.1.3.1.56.1	
dtad dead	Section 2.1.5.4.15	
sound dead	Section 2.1.5.4.16	
current temperature	Section 2.1.4.2.1.6.1	
current plus12	Section 2.1.4.2.1.4.1	
plus12 dead	Section 2.1.5.4.17	
current minus12	Section 2.1.4.2.1.3.1	
minus12 dead	Section 2.1.5.4.18	
current plus5	Section 2.1.4.2.1.5.1	
plus5 dead	Section 2.1.5.4.19	
status out	Section 2.1.5.1.13	

Table 2.1-533: monitor_status Information.

2.1.6 Keyboard Interface Software

The C archive library libkeybrd provides a device-independent interface to read characters from the host keyboard. This interface is used for development /debugging. Once each frame, the simulation host checks if a key has been pressed. If it has, the key-press is interpreted by vehicle specific routines in m1_keybrd.c, m2_keybrd.c, and kato_keybrd.c. The structure of this CSC is depicted in Figure 2.1-7. This functionality is realized by the following CSUs:

libkeybrd
m1_keybrd.c
m2_keybrd.c
kato_keybrd.c

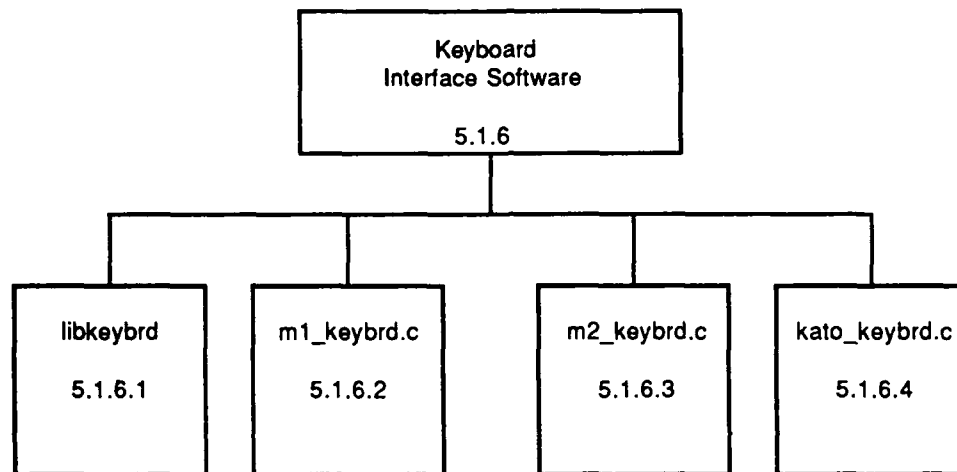


Figure 2.1-7 Keyboard Interface Software structure.

2.1.6.1 libkeybrd

(./simnet/release/src/libsrc/libkeybrd [libkeybrd])

Libkeybrd contains the routines for maintaining a machine independent interface for the terminal keyboard.

2.1.6.1.1 close.c

(./simnet/release/src/libsrc/libkeybrd/close.c)

This file contains the routine for closing a libkeybrd type tty connection.

Includes:

"stdio.h" (Simnet Butterfly Machines only)
"key_loc.h"

2.1.6.1.1.1 keybrd_tty_close

This routine closes the terminal keyboard.

Parameters		
Parameter	Type	Where Typedef Declared
desc	int	Standard

Table 2.1-534: keybrd_tty_close Information.

2.1.6.1.2 init.c

(./simnet/release/src/libsrc/libkeybrd/init.c)

This file contains routines for initializing raw tty inputs.

If Simnet Butterfly Machine, includes:

"stdio.h"
 "fnctl.h"
 "key_loc.h"

If Masscomp Machine, includes:

"fnctl.h"
 "sys/ioctl.h"
 "termio.h"
 "key_loc.h"

2.1.6.1.2.1 keybrd_tty_init

This routine does the machine hardware dependent operations and returns a raw file descriptor for use by the application in referencing the tty (in read, write, and close functions). Note that tty 0 is assumed to be the Simulation console.

Parameters		
Parameter	Type	Where Typedef Declared
tty	int	Standard
mode	int	Standard
Internal Variables		
Variable	Type	Where Typedef Declared
fd	int	Standard
stio	struct termio	Standard
device_name[12]	char	Standard
Return Values		
Return Value	Type	Meaning
fd	int	a handle for subsequent calls to libkeybrd

Table 2.1-535: keybrd_tty_init Information for Masscomp Machine.

Parameters		
Parameter	Type	Where Typedef Declared
tty	int	Standard
mode	int	Standard
Return Values		
Return Value	Type	Meaning
tty	int	a handle for subsequent calls to libkeybrd
Calls		
Function	Where Described	
ChannelHasInput		

Table 2.1-536: keybrd_tty_init Information for Simnet Butterfly Machine.

2.1.6.1.3 key_loc.h (./simnet/release/src/libsrc/libkeybrd/key_loc.h)

This file contains definitions needed only in libkeybrd.

Defines:

USE_CONSOLE
USE_HOSTIN (for Butterfly Machine)

2.1.6.1.4 read.c

(./simnet/release/src/libsrc/libkeybrd/read.c)

Includes:

"key_loc.h"

"stdio.h" (for Simnet Butterfly Machine)

2.1.6.1.4.1 keybrd_tty_read

This routine reads one character from the terminal keyboard.

Parameters		
Parameter	Type	Where Typedef Declared
desc	int	Standard
Internal Variables		
Variable	Type	Where Typedef Declared
key	char	Standard
Return Values		
Return Value	Type	Meaning
key	char	the next character in the buffer
0	int	no character is in the buffer

Table 2.1-537: keybrd_tty_read Information for Masscomp Machine.

Parameters		
Parameter	Type	Where Typedef Declared
desc	int	Standard
Internal Variables		
Variable	Type	Where Typedef Declared
key	char	Standard
Return Values		
Return Value	Type	Meaning
key	char	the next character in the buffer
0		no character is in the buffer

Table 2.1-538: keybrd_tty_read Information for Simnet Butterfly Machine.

2.1.6.1.5 reset.c

(./simnet/release/src/libsrc/libkeybrd/reset.c)

Includes:

"key_loc.h"

"sys/ioctl.h" (for Masscomp Machine)

"termio.h" (for Masscomp Machine)

"stdio.h" (for Butterfly Machine)

2.1.6.1.5.1 keybrd_tty_reset

This routine resets the terminal keyboard back to normal parameters after use by the simulation.

Parameters		
Parameter	Type	Where Typedef Declared
desc	int	Standard
Internal Variables		
Variable	Type	Where Typedef Declared
stio	struct termio	Standard

Table 2.1-539: keybrd_tty_reset Information for Masscomp Machine.

Parameters		
Parameter	Type	Where Typedef Declared
desc	int	Standard

Table 2.1-540: keybrd_tty_reset Information for Butterfly Machine.

2.1.6.3.7 keyboard_exit_gracefully

This routine resets the keyboard and closes the connection.

Calls	
Function	Where Described
keyboard_reset_terminal	Section 2.1.6.3.6

Table 2.1-551: keyboard_exit_gracefully Information.

2.1.6.1.6 write.c

Includes:

"key_loc.h"

"stdio.h" (for Simnet Butterfly Machine only)

2.1.6.1.6.1 keybrd_tty_write

This routine writes characters to the terminal.

Parameters		
Parameter	Type	Where Typedef Declared
desc	int	Standard
data	pointer to char	Standard
size	int	Standard
Return Values		
Return Value	Type	Meaning
write(des,data,size)	int	For Masscomp Machine Only: writes to terminal.
-1	int	For Butterfly Machine: not supported.

Table 2.1-541: keybrd_tty_write Information.

2.1.6.2 m1_keybrd.c

(/simnet/release/src/vehicle/m1/src/m1_keybrd.c [m1_keybrd.c])

This file contains the M1 specific routines required to interface with the terminal keyboard.

Includes:

```
"stdio.h"
"fcntl.h"
"signal.h"          if SIMBFLY not defined
"termio.h"          if SIMBFLY not defined
"sim_dfns.h"
"sim_types.h"
"sim_macros.h"
"rtc.h"
"pro_sim.h"
"pro_data.h"
"repair_m1.h"
"mass_stdh.h"
"dgi_stdg.h"
"sim_cig_if.h"
"libfail.h"
"librepair.h"
"libmem_dfn.h"
"libnetwork.h"
"timers.h"
"libsound.h"
"libhull.h"
"libkin.h"
"libfilter.h"
"librva.h"
"m1_main.h"
"m1_cntrl.h"
"m1_elecsys.h"
"m1_ammo_pn.h"
"m1_engine.h"
"m1_laser.h"
"m1_fuelsys.h"
"m1_weapons.h"
"m1_tracks.h"
"m1_turret.h"
"m1_dtrain.h"
"m1_vision.h"
"m1_repair.h"
"m1_sound.h"
"m1_ammo.h"
"m1_keybrd.h"
"m1_cupola.h"
"status.h"
"m1_status.h"
"m1_turr_pn.h"
"m1_driv_pn.h"
"net/network.h"
"enpioctl.h"        if SIMBFLY not defined
```

Includes:
 "enpsvr.h" if SIMBFLY defined

Defines:

<u>Symbol</u>	<u>Value</u>
DELTA_POT	0.005

int variable declarations and initialization:

```
console
use_keyboard = FALSE
use_cupola = FALSE
```

REAL variable declarations and initialization:

```
lpscope_value = 0.0
cws_value = 0.0
```

Pointers to REAL typedef declarations:
 vec

Procedure declarations:
 keyboard_setup_terminal()

2.1.6.2.1 keyboard_really_use

This routine enables the keyboard.

2.1.6.2.2 keyboard_use_cupola

This routine sets the variable *use_cupola* to TRUE and displays a message that cupola and periscope are now under keyboard control.

2.1.6.2.3 keyboard_init

This routine initializes the keyboard.

Calls	
Function	Where Described
keyboard_setup_terminal	Section 2.1.6.2.5
cupola_lpscope_new_value	Section 2.3.6.1
cupola_cws_new_value	Section 2.3.6.1.2.4

Table 2.1-542: keyboard_init Information.

2.1.6.2.4 keyboard_simul

If SIMBFLY is defined, the long variable *pkt_cnt_start* is declared prior to this routine. This routine takes the appropriate action for a key hit by a user.

Internal Variables		
Internal Variable	Type	Where Typedef Declared
cmd	char	Standard
network_stats[N_STATS]	int	Standard
network_statstr[41]	char	Standard
n	int	Standard
Errors		
Error Name	Reason for Error	
KEYBOARD: can't get network statistics	A call to net_print_statistics returned -1.	
KEYBOARD: can't zero network statistics	A call to net_zero_statistics returned -1.	
Calls		
Function	Where Described	
keybrd tty read	Section 2.1.6.1.4.1	
fail break system	Section 2.5.4.8.1	
reconstitute from keyboard	Section 2.1.1.3.2.18.7	
repair stop repair		
controls break controls	Section 2.2.2	
controls restore controls	Section 2.2.2	
deactivate simulation	Section 2.5.1.1.4	
filter_dump filter info	Section 2.5.14.4.1	
repair fix system	Section 2.5.4.19.3	
network get vehicle force	Section 2.1.1.3.1.17.1	
map set asid debug		
electsys battery failure	Section 2.2.6.3.1.17	
rva dump priority lists	Section 2.5.12.2.3	
map print		
rva turn debug on	Section 2.5.12.2.1	
rva turn debug off	Section 2.5.12.2.2	
vision break all blocks	Section 2.3.6.3.2.9	
exit gracefully	Section 2.5.1.1.2	
controls kill radio	Section 2.2.2	
controls restore radio	Section 2.2.2	
use static debug	Section 2.1.2.2.2.116.1	
controls electsys dead	Section 2.2.2	
controls electsys reborn	Section 2.2.2	
sound reset		
fail cat kill	Section 2.5.4.9.1	
fuel init tanks	Section 2.2.5.2.1	
repair all systems	Section 2.5.4.19.7	
cupola cws new value	Section 2.2.6.1.2.4	
network_print statistics	Section 2.1.1.3.2.16.1	
status_print_temp_and_supplies	Section 2.1.5.2.8	

toggle_gunner_vision_state	Section 2.2.6.4.3.33
toggle_driver_vision_state	Section 2.2.6.4.3.32
print_view_modes	Section 2.2.6.4.3.34
HELP_PRINT1	sim_macros.h
HELP_PRINT2	sim_macros.h
ammo_restore_ammo	Section 2.2.5.1.65
controls_restore_ammo	Section 2.2.2
timers_status	Section 2.6.3.17.1
net_print_statistics	
nprintf	Section 2.1.1.3.1.34.1
net_zero_statistics	
turret_send_azimuth_ind	Section 2.5.5.2.18
turret_null_azimuth_ind	Section 2.5.5.2.17
bdd_rtc_statistics	
network_get_vehicle_id	Section 2.1.1.3.1.22.1
network_get_exercise_id	Section 2.1.1.3.1.16.1
kinematics_get_o_to_h	Section 2.5.8.1.4
ammo_print_statistics	Section 2.2.5.1.69
print_reasons	Section 2.5.16.3.6
vision_cmds_binoculars	Section 2.2.6.4.3.6
get_ballistics_debug	Section 2.1.2.2.6.2
set_ballistics_debug	Section 2.1.2.2.6.1
rtc_print_permanent	Section 2.6.16.1.11

Table 2.1-543: keyboard_simul Information.

2.1.6.2.5 keyboard_setup_terminal

This routine sets up the keyboard to be read by an application.

Calls	
Function	Where Described
keybrd_tty_init	Section 2.1.6.1.2.1

Table 2.1-544: keyboard_setup_terminal Information.

2.1.6.2.6 keyboard_reset_terminal

This routine resets the keyboard before exiting the application.

Calls	
Function	Where Described
keybrd_tty_reset	Section 2.1.6.1.5.1

Table 2.1-545: keyboard_reset_terminal Information.

2.1.6.2.7 keyboard_exit_gracefully

If the keyboard was enabled, this routine resets the keyboard and closes the connection.

Calls	
Function	Where Described
keyboard_reset_terminal	Section 2.1.6.2.6
keybrd_tty_close	Section 2.1.6.1.1.1

Table 2.1-546: keyboard_exit_gracefully Information.

2.1.6.3 m2_keybrd.c

(/simnet/release/src/vehicle/m2/src/m2_keybrd.c [m2_keybrd.c])

This file contains the M2 specific routines required to interface with the terminal keyboard.

Includes:

```
"stdio.h"
"fcntl.h"
"signal.h"          if SIMBFLY not defined
"termio.h"          if SIMBFLY not defined
"sim_dfns.h"
"sim_types.h"
"rtc.h"
"pro_sim.h"
"pro_data.h"
"mass_stdc.h"
"dgi_stdg.h"
"sim_cig_if.h"
"libcig.h"
"libfail.h"
"librepair.h"
"libmain.h"
"libmem_dfn.h"
"libnetwork.h"
"timers.h"
"libsound.h"
"libhull.h"
"libkin.h"
"librva.h"
"m2_cntrl.h"
"m2_turret.h"
"m2_sound.h"
"m2_keybrd.h"
"m2_dtrain.h"
"m2_cupola.h"
"status.h"
"m2_status.h"
"net/network.h"
"m2_fuelsys.h"
"m2_cig.h"
"m2_isu.h"
"m2_firectl.h"
"m2_ammo.h"
"m2_weapons.h"
"m2_main.h"
"m2_vision.h"
"m2_driv_pn.h"
"m2_turr_pn.h"
"fifo_dfn.h"
"m2_mem_dfn.h"
"fifo.h"
"enpioctl.h"        if SIMBFLY not defined
"enpsvr.h"          if SIMBFLY defined
```

Defines:

<u>Symbol</u>	<u>Value</u>
MAXFIELDS	32
DELTA_POT	0.005

int variable declarations and initialization:

```
mask
use_keyboard = FALSE
use_cupola = FALSE
```

REAL variable declarations and initialization:

```
cws_value = 0.0
```

Pointers to REAL typedef declarations:

```
vec
```

Procedure declarations:

```
keyboard_setup_terminal()
```

2.1.6.3.1 keyboard_really_use

This routine enables the keyboard.

2.1.6.3.2 keyboard_use_cupola

This routine sets the variable *use_cupola* to TRUE and displays a message that cupola is now under keyboard control.

2.1.6.3.3 keyboard_init

This routine initializes the keyboard.

Calls	
Function	Where Described
keyboard_setup_terminal	Section 2.1.6.3.5
cupola_cws_new_value	Section 2.3.6.1.2.4

Table 2.1-547: keyboard_init Information.

2.1.6.3.4 keyboard_simul

This routine takes the appropriate action for a key hit by a user.

Internal Variables		
Internal Variable	Type	Where Typedef Declared
cmd	char	Standard
local_mask	int	Standard
network_stats[N_STATS]	int	Standard
network_statstr[41]	char	Standard
n	int	Standard
list	pointer to pointer to RVA_ENTRY	librva.h
num_vehs	int	Standard
Errors		
Error Name	Reason for Error	
KEYBOARD: can't get network statistics	A call to net_print_statistics returned -1.	
KEYBOARD: can't zero network statistics	A call to net_zero_statistics returned -1.	
Calls		
Function	Where Described	
ammo print ammo variables	Section 2.3.5.1.68	
reconstitute from: keyboard	Section 2.1.1.3.2.18.7	
drivetrain parking brake set	Section 2.3.6.2.4.12	
drivetrain_parking_brake_ release	Section 2.3.6.2.4.13	
controls break controls	Section 2.3.2	
controls restore controls	Section 2.3.2	
filter dump filter info	Section 2.5.14.4.1	
cig gps mag 4x	Section 2.3.6.3.2.4	
firectl gps mag 4x	Section 2.3.2.2.3.3	
isu gps mag 4x	Section 2.3.6.3.3.4	
ammo gps mag 4x	Section 2.3.5.1.45	
cig gps mag 12x	Section 2.3.6.3.2.3	
firectl gps mag 12x	Section 2.3.2.2.3.4	
isu gps mag 12x	Section 2.3.6.3.3.3	
ammo gps mag 12x	Section 2.3.5.1.44	
transmission dump		
drivetrain set brake	Section 2.3.6.2.4.11	
rva dump priority lists	Section 2.5.12.2.3	
silent_mode on	Section 2.3.1.1.1	
silent_mode off	Section 2.3.1.1.2	
rva turn debug on	Section 2.5.12.2.1	
rva turn debug off	Section 2.5.12.2.2	
exit gracefully	Section 2.5.1.1.2	
resupply gating conditions	Section 2.3.5.3.15	
ammo_ready_to_external_ resupply	Section 2.3.5.1.70	

keybrd_ammo_carriers_near_here	Section 2.3.5.3.35
use static debug	Section 2.1.2.2.2.116.1
controls receive flash	Section 2.3.2
controls receive off	Section 2.3.2
weapons keybrd fire	Section 2.3.3.2.21
fail cat kill	Section 2.5.4.9.1
fuel init tanks	Section 2.2.5.2.1
repair all systems	Section 2.5.4.19.7
controls electsys reborn	Section 2.3.2
cupola cws new value	Section 2.3.6.1.2.4
network print statistics	Section 2.1.1.3.2.16.1
status_print_temp_and_supplies	Section 2.1.5.2.8
ammo restore ammo	Section 2.2.5.1.65
timers_status	Section 2.6.3.17.1
net_print_statistics	
nprintf	Section 2.1.1.3.1.34.1
net_zero_statistics	Section
print br values	Section 2.3.6.3.2.24
get ballistics debug	Section 2.1.2.2.6.2
set ballistics debug	Section 2.1.2.2.6.1
ammo round indexed status	Section 2.3.5.1.28
kinematics_get_o_to_h	Section 2.5.8.1.4
cig_get_db	Section 2.2.1.33.1
rva_get_close_list	Section 2.5.12.9.1
print reasons	Section 2.5.16.3.6

Table 2.1-548: keyboard_simul Information.

2.1.6.3.5 keyboard_setup_terminal

This routine sets up the keyboard to be read by an application.

Internal Variables		
Internal Variable	Type	Where Typedef Declared
fd	int	Standard
stio	struct termio	termio.h

Table 2.1-549: keyboard_setup_terminal Information.

2.1.6.3.6 keyboard_reset_terminal

This routine resets the terminal before exiting the application.

Internal Variables		
Internal Variable	Type	Where Typedef Declared
fd	int	Standard
stio	struct termio	termio.h

Table 2.1-550: keyboard_reset_terminal Information.

2.1.6.4 kato keybrd.c

(/simnet/release/src/vehicle/kato/src/kato_keybrd.c [kato_keybrd.c])

This file contains the Stealth vehicle specific routines required to interface with the terminal keyboard.

Includes:

```

"stdio.h"
"fcntl.h"
"sim_dfns.h"
"sim_types.h"
"rtc.h"
"pro_sim.h"
"pro_data.h"
"mass_stdh.h"
"dgi_stdg.h"
"sim_cig_if.h"
"libfail.h"
"librepair.h"
"libmem_dfn.h"
"libnetwork.h"
"timers.h"
"libsound.h"
"libhull.h"
"libkin.h"
"librva.h"
"kato_cntrl.h"
"kato_keybrd.h"
"status.h"
"kato_status.h"
"kato_sound.h"
"net/network.h"
"kato_cntrlr.h"
"kato_state.h"
"kato_view.h"
"enpioctl.h"          if SIMBFLY not defined
"enpsvr.h"            if SIMBFLY defined

```

Defines:

<u>Symbol</u>	<u>Value</u>
TTY_CONSOLE	0

int variable declarations and initialization:

```

console_desc
use_keyboard = FALSE

```

Pointers to REAL typedef declarations:

```

vec

```

Procedure declarations:

```

keyboard_setup_terminal()

```

2.1.6.4.1 keyboard_really_use

This routine enables the keyboard.

2.1.6.4.2 keyboard_init

This routine initializes the keyboard.

Calls	
Function	Where Described
keyboard setup terminal	Section 2.1.6.4.5

Table 2.1-552: keyboard_init Information.

2.1.6.4.3 keyboard_simul

This routine takes the appropriate action when a user hits a key.

Internal Variables		
Internal Variable	Type	Where Typedef Declared
cmd	char	Standard
n	int	Standard
Errors		
Error Name	Reason for Error	
KEYBOARD: can't get network statistics	A call to net_print_statistics returned -1.	
KEYBOARD: can't zero network statistics	A call to net_zero_statistics returned -1.	
Calls		
Function	Where Described	
keybrd tty read	Section 2.1.6.1.4.1	
view get pitch angle	Section 2.1.2.2.9.13	
controls break controls	Section 2.4.2	
controls restore controls	Section 2.4.2	
state vehicle detach	Section 2.4.8.2	
filter dump filter info	Section 2.5.14.4.1	
view centered	Section 2.1.2.2.9.6	
state toggle fix	Section 2.4.8.2	
rva dump priority lists	Section 2.5.12.2.3	
state mimic	Section 2.4.8.2	
rva turn debug on	Section 2.5.12.2.1	
rva turn debug off	Section 2.5.12.2.2	
exit gracefully	Section 2.5.1.1.2	
vehicle restart	Section 2.5.19.1.7	
state saf mode on	Section 2.4.8.2	
state saf mode off	Section 2.4.8.2	
state terrain follow on	Section 2.4.8.2	
state terrain follow off	Section 2.4.8.2	
state vel attach	Section 2.4.8.2	
state world attach	Section 2.4.8.2	
state orbit attach	Section 2.4.8.2	
veh set force	Section 2.6.10.6.2	
network set force	Section 2.1.1.3.1.50.1	
network print statistics	Section 2.1.1.3.2.16.1	
status_print_temp_and_supplies	Section 2.1.5.2.8	
timers status	Section 2.6.3.17.1	
net print statistics		
nprintf	Section 2.1.1.3.1.34.1	
kinematics get o to h	Section 2.5.8.1.4	
get n mapped	Section 2.1.2.2.2.34.1	
print reasons	Section 2.5.16.3.6	

Table 2.1-553: keyboard_simul Information.

2.1.6.4.4 keyboard_setup_terminal

This routine sets the keyboard to be read by an application.

Calls	
Function	Where Described
keybrd_tty_init	Section 2.1.6.1.2.1

Table 2.1-554: keyboard_setup_terminal Information.

2.1.6.4.5 keyboard_reset_terminal

This routine resets the keyboard before exiting the application.

Calls	
Function	Where Described
keybrd_tty_reset	Section 2.1.6.1.5.1

Table 2.1-555: keyboard_reset_terminal Information.

2.1.6.4.6 keyboard_exit_gracefully

This routine resets the keyboard and closes the connection.

Calls	
Function	Where Described
keyboard_reset_terminal	Section 2.1.6.4.6
keybrd_tty_close	Section 2.1.6.1.1.1

Table 2.1-556: keyboard_exit_gracefully Information.

2.1.7 Spaceball Interface Software

Figure 2.1-8 shows the structure of this CSC.

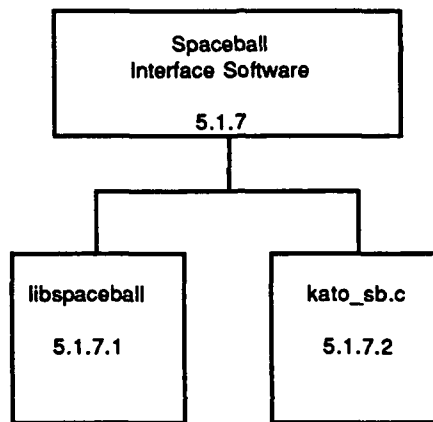


Figure 2.1-8: Spaceball Interface Software structure.

Two CSUs are associated with the Spaceball Interface. They are libspaceball and kato_sb.c.

2.1.7.1 libspaceball

(/simnet/release/src/vehicle/libsrc/libspaceball [libspaceball])

Code contained in the libspaceball files is licensed for use and distribution by Spatial Systems, Inc. Refer to the Spaceball Application Developer's Reference (Spatial Systems, Inc. Document Number D1005) for more information.

2.1.7.1.1 sbcustom.c

(/simnet/release/src/vehicle/libsrc/libspaceball/sbcustom.c)

Spaceball Library RS232-C I/O and Clock Routines for UNIX

The Spaceball Library interfaces with the operating system via the routines contained in this file:

SbOpen()	-- open the Spaceball tty
SbClose()	-- close the currently open Spaceball tty
SbInput()	-- read in characters from Spaceball
SbOutput()	-- send characters to Spaceball
SbMilliseconds()	-- returns the lowest 16 bits of a millisecond clock

2.1.7.1.2 sblibry.c

(/simnet/release/src/vehicle/libsrc/libspaceball/sblibry.c)

Spaceball Library Routines

This file provides the functionality to convert between Spaceball data formats and application data formats. Every effort has been made to allow simple configuration parameters to be set in "sbcustom.h" so as to meet the requirements of a wide variety of applications. The use of the routines contained in this file is strongly recommended to save implementation time and effort and to ensure portability.

2.1.7.1.3 sbtest.c

(/simnet/release/src/vehicle/libsrc/libspaceball/sbtest.c)

Spaceball Library Test Program

This file provides a simple test program to verify the operation of Spaceball.

2.1.7.1.4 sbtute.c

(/simnet/release/src/vehicle/libsrc/libspaceball/sbtute.c)

Spaceball Library Tutorial Example

This file contains template routines which can be used and built upon by application programmers. It is written in a tutorial fashion.

2.1.7.1.5 sbcustom.h

(/simnet/release/src/vehicle/libsrc/libspaceball/sbcustom.h)

This file is the configuration file for the Spaceball Library Unix implementation.

2.1.7.1.6 sblibry.h

(/simnet/release/src/vehicle/libsrc/libspaceball/sblibry.h)

Spaceball Library Include File

This file provides the definitions and declarations required to use the Spaceball Library and should be included in each program utilizing the library.

2.1.7.2 kato_sb.c

(/simnet/release/src/vehicle/kato/src/kato_sb.c [kato_sb.c])

Includes:

```
"stdio.h"
"simstdio.h"
"sblibry.h"
"basic.h"
"sim_types.h"
"sim_macros.h"
"kato_state.h"
```

Defines:

Sb_ENVIRONMENT_INCLUDES

ForceID declaration and initialization:

force = otherForceID

int variable declarations and initialization:

```
tran_on = TRUE
rot_on = TRUE
dominant = FALSE
```

SbUInt16 declaration:

b_period

float declarations and initialization:

```
trans_gain_factor = 0.2
rot_gain_factor = 0.2
trans_gain = 3.0
rot_gain = 5.0
tran_vec[3]
rot_vec[3]
```

2.1.7.2.1 spaceball_simul

Internal Variables		
Internal Variable	Type	Where Typedef Declared
pkt[200]	char	Standard
now	SbUInt16	COTS Spaceball software
Calls		
Function	Where Described	
SbCheckSpaceball	COTS Spaceball software	
SbRequestBallData	COTS Spaceball software	

Table 2.1-557: spaceball_simul Information.

2.1.7.2.2 spaceball_exit

Calls	
Function	Where Described
SbSetBallMode	COTS Spaceball software
SbClose	COTS Spaceball software

Table 2.1-558: spaceball_exit Information.**2.1.7.2.3 initialize_spaceball**

Parameters		
Parameter	Type	Where Typedef Declared
tty_line	pointer to char	Standard
Errors		
Error Name	Reason for Error	
Could not open Spaceball on ...	A call to SbOpen returned a value < 0.	
Spaceball did not appear to respond on ... - continuing anyway	SbReset(TRUE) returned -1.	
Calls		
Function	Where Described	
SbOpen	COTS Spaceball software	
SbReset	COTS Spaceball software	
SbSetBallMode	COTS Spaceball software	
SbRezero	COTS Spaceball software	
SbSetTranslationFeel	COTS Spaceball software	
SbSetRotationFeel	COTS Spaceball software	
SbSetNullRegion	COTS Spaceball software	
SbRequestBallData	COTS Spaceball software	

Table 2.1-559: initialize_spaceball Information.**2.1.7.2.4 display_data**

Internal Variables		
Internal Variable	Type	Where Typedef Declared
now	SbUInt16	COTS Spaceball software
Calls		
Function	Where Described	
SbMilliseconds	COTS Spaceball software	

Table 2.1-560: display_data Information.

2.1.7.2.5 mypressed

Parameters		
Parameter	Type	Where Typedef Declared
keys	SbUInt16	COTS Spaceball software
Calls		
Function	Where Described	
SbBeep	COTS Spaceball software	
SbSetBallMode	COTS Spaceball software	
handles_pil_thumb_lower_depressed	Section 2.4.2	
veh set force	Section 2.6.10.6.2	
network set force	Section 2.1.1.3.1.50.1	
SbRezero	COTS Spaceball software	
handles_pil_trigger_1_depressed	Section 2.4.2	

Table 2.1-561: mypressed Information.

2.1.7.2.6 mytranslate

Parameters		
Parameter	Type	Where Typedef Declared
period	SbUInt16	COTS Spaceball software
vec[3]	SbReal	COTS Spaceball software
Internal Variables		
Internal Variable	Type	Where Typedef Declared
translations	VECTOR	sim types.h
i	int	Standard
Calls		
Function	Where Described	
SbDominant	COTS Spaceball software	
min	sim macros.h	
max	sim macros.h	
controller main cyclic long	Section 2.4.8.1	
controller main cyclic lateral	Section 2.4.8.1	
controller main collective	Section 2.4.8.1	

Table 2.1-562: mytranslate Information.

2.1.7.2.7 myrotate

Parameters		
Parameter	Type	Where Typedef Declared
period	SbUint16	COTS Spaceball software
vec[3]	SbReal	COTS Spaceball software
Internal Variables		
Internal Variable	Type	Where Typedef Declared
rotations	VECTOR	sim_types.h
pitch rate	REAL	sim_types.h
i	int	Standard
Calls		
Function	Where Described	
SbDominant	COTS Spaceball software	
min	sim_macros.h	
max	sim_macros.h	
pedal tail rotor collective	Section 2.4	
state return attach mode	Section 2.4.8.2	
state return attach state	Section 2.4.8.2	
sign	sim_macros.h	
view set pitch rate	Section 2.1.2.2.9.5	

Table 2.1-563: myrotate Information.

2.2 M1 Vehicle Simulation Functions

A distributed vehicle simulation performs a set of tasks each frame. It checks the state of its controls, updates lights and meters, checks its failure/repair model, manages munitions, models its hull (and turret) subsystems, does kinematics/dynamics, simulates weapons, and communicates with the outside world. This cycle of tasks is referred to below as the main simulation loop.

The structure of the M1 Vehicle Simulation Software CSC is shown in Figure 2.2-1.

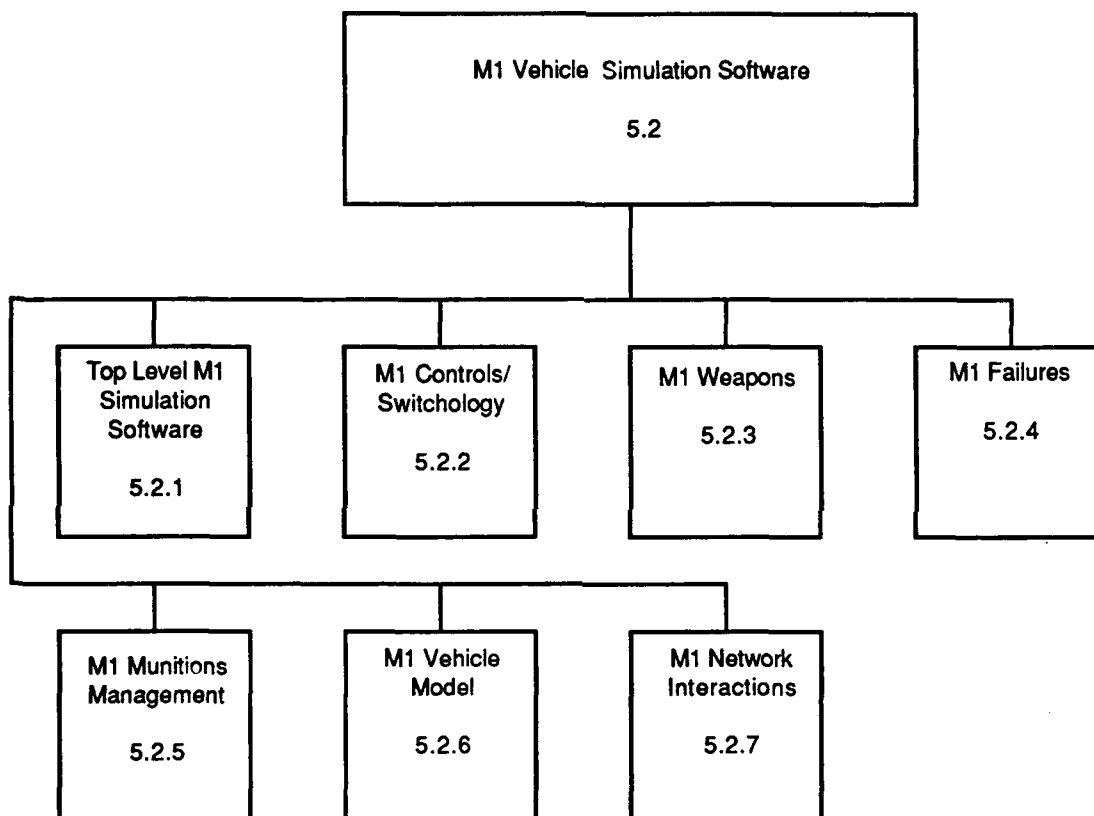


Figure 2.2-1: M1 Vehicle Simulation Software CSC structure.

The second level CSC's that make up this CSC are as follows:

- Top Level M1 Simulation Software
- M1 Controls / Switchology
- M1 Weapons
- M1 Failures
- M1 Munitions Management
- M1 Vehicle Model
- M1 Network Interactions

2.2.1 Top Level M1 Simulation Software

The main vehicle simulation loop is executed once each frame. Vehicle specific simulation routines are called by the one csu, m1_main.c.

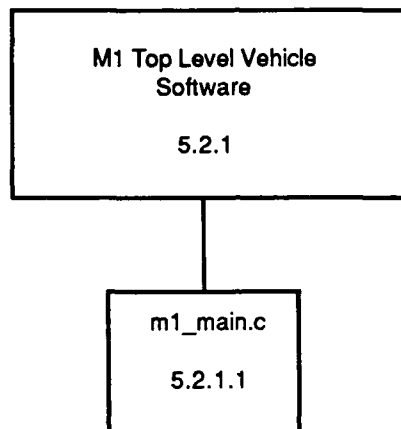


Figure 2.2-2: M1 Top Level Vehicle Software CSC structure.

2.2.1.1 m1_main.c

(/simnet/release/src/vehicle/m1/src/m1_main.c [m1_main.c])

This file contains routines used in the SIMNET simulation of the M1 Abrams Main Battle Tank.

Includes:

"stdio.h"	"ctype.h"
"signal.h"	"sim_dfns.h"
"sim_macros.h"	"sim_types.h"
"mass_std.c.h"	"dgi_stdg.h"
"sim_cig_if.h"	"pro_assoc.h"
"pro_sim.h"	"status.h"
"status_m1.h"	"veh_type.h"
"fifo_dfn.h"	"fifo.h"
"bigwheel.h"	"libterrain.h"
"libkin.h"	"libfail.h"
"libcig.h"	"libms_.h"
"bbd.h"	"libhull.h"
"libidc.h"	"libmain.h"
"libmem.h"	"libnetwork.h"
"librepair.h"	"librva.h"
"libsusp.h"	"libturret.h"
"libsound.h"	"libmap.h"
"m1_ammo.h"	"m1_bcs.h"
"m1_cntrl.h"	"m1_cupola.h"
"m1_dtrain.h"	"m1_elecsys.h"
"m1_failure.h"	"m1_firectl.h"
"m1_fuelsys.h"	"m1_handles.h"
"m1_hydrsys.h"	"m1_keybrd.h"

"m1_laser.h"	"m1_main.h"
m1_meter.h"	"m1_pots.h"
"m1_repair.h"	"m1_resupp.h"
"m1_sound.h"	"m1_weapons.h"
"m1_turret.h"	"m1_vision.h"
"m1_status.h"	"m1_thermal.h"
"timers.h"	"dtad.h"
"status.h"	"ser_status.h"
"cmc.h"	"cmc_timers.h"
"cmc_status.h"	

The following are declared:

```

debug
print_overruns
guise_override
guise_to_use
exit()

```

The following are declared for the '-p' switch:

```

init_activ
initial_activation

```

The following is defined:

```

PARS_FILE

```

2.2.1.1.1 print_help

This routine prints out data for the M1 simulation.

Parameters		
Parameter	Type	Where Typedef Declared
progname	pointer to char	Standard

Table 2.2-1: print_help Information.

2.2.1.1.2 print_veh_logo

This routine prints a logo for the M1 vehicle.

2.2.1.1.3 veh_spec_startup

This routine sets up the network and CIG interfaces at startup. In addition, the simulator type is set and the damage tables are initialized.

Calls	
Function	Where Described
rtc init clock	Section 2.6.16.1.14
network set simulator type	Section 2.1.1.3.1.53.1
use cig reconfig startup	
cig set view config file	Section 2.1. 2.2.2.23.1
get vconfig file1	Section 2.5.1.2.2
map vehicle file read	Section 2.6.11.5.1
get veh map file	Section 2.5.1.2.5
map read asid file	Section 2.6.11.4.1
get asid map file	Section 2.5.1.2.4
map file read	Section 2.6.11.3.1
get ammo map file	Section 2.5.1.2.6
keyboard init	Section 2.1.6.2
failure init	Section 2.2.4.1.1
map_get_damage_files	Section 2.6.11.1.1

Table 2.2-2: veh_spec_startup Information.

2.2.1.1.4 veh_spec_idle

This routine is called while the simulator is in the IDLE state.

Calls	
Function	Where Described
status simul	Section 2.1.5.2
keybrd simul	Section 2.1.6.2
io simul idle	Section 2.1.2.2.5.1.2
process activate request	Section 2.1.1.3.2.1.1
network get exercise id	Section 2.1.1.3.1.16.1

Table 2.2-3: veh_spec_idle Information.

2.2.1.1.5 veh_spec_init

Order dependent initializations are performed for all of the M1's subsystems while the simulator is in the SIMINIT state.

Calls	
Function	Where Described
cupola_init	Section 2.2.6.1.2
sound_init	Section 2.1.3.2.4
status_preset	Section 2.1.5.2
controls_fsm_init	Section 2.2.2
resupply_init	Section 2.2.5.3.28
meter_init	Section 2.2.2.3.1
electsys_init	Section 2.2.6.3.17
hydraulic_init	Section 2.2.6.4.2
firectl_init	Section 2.2.2.2.3
drivetrain_init	Section 2.2.6.2.1.41
handles_init	Section 2.2.2.2.2
laser_init	Section 2.2.3.2
bcs_init	Section 2.2.3.1
weapons_init	Section 2.2.3.3
vision_restore_all_blocks	Section 2.2.6.4.3
controls_edge_init	Section 2.2.2
app_init	Section 2.2.7.1.5
thermal_init	Section 2.2.6.4.4
config_pos_init2	Section 2.1.2.2.2.24.2
kinematics_get_o_to_h	Section 2.5.8.2.4
kinematics_get_w_to_h	Section 2.5.8.2.1
cig_init_ctr	Section 2.1.2.2.6.3

Table 2.2-4: veh_spec_init Information.

2.2.1.1.6 veh_spec_simulate

This routine calls the routines which simulate the various functions of the M1's subsystems on a tick by tick basis.

Internal Variables		
Internal Variable	Type	Where Typedef Declared
start	long	Standard
end	long	Standard
Calls		
Function	Where Described	
status simul	Section 2.1.5.2	
keyboard simul	Section 2.1.6.2	
sound simul	Section 2.1.3.2.6	
controls simul	Section 2.2.2	
handles simul	Section 2.2.2.2.2	
ammo simul	Section 2.2.5.11.2	
resupply simul	Section 2.2.5.3.29	
electsys simul	Section 2.2.6.3.1.1	
hydraulic simul	Section 2.2.6.4.2.1	
fuel simul	Section 2.2.5.2.3	
drivetrain simul	Section 2.2.6.2.1.40	
bcs simul	Section 2.2.3.1	
weapons simul	Section 2.2.3.3	
cupola simul	Section 2.2.6.1.2	
thermal simul	Section 2.2.6.4.4	

Table 2.2-5: veh_spec_simulate Information.

2.2.1.1.7 veh_spec_stop

This routine is called while the simulator is in the SIMSTOP state. The IDC and sound system hardware is reinitialized, and the vision blocks are broken (all screens are blackened).

Calls	
Function	Where Described
idc init	Section 2.1.4.1.1.24.1
sound init	Section 2.1.3.2.4
vision break all blocks	Section 2.2.6.4.3

Table 2.2-6: veh_spec_stop Information.

2.2.1.1.8 veh_spec_exit

This routine is called while the simulator is in the SIMEXIT state. Simulation statistics are printed, and the network connection is closed.

Internal Variables		
Internal Variable	Type	Where Typedef Declared
num_ticks	int	Standard
Calls		
Function	Where Described	
keyboard_exit_gracefully	Section 2.1.6.2	
timers_get_current_time	Section 2.6.3.2.1	
timers_get_current_tick	Section 2.6.3.1.1	
timers_elapsed_milliseconds	Section 2.6.3.10.1	
network_print_statistics	Section 2.1.1.3.2.16.1	
net_close	Section 2.20.2.3.1 in MCC CSCI SDD	

Table 2.2-7: veh_spec_exit Information.

2.2.1.1.9 main

This routine loops through the M1 simulation once each frame. The generic simulation routines in "libmain" is called by this routine.

The parameters are read in and parsed:

case -a	The request send and receive sizes are set and assymetric is set to on.
case -A	The ammo autoloader is enabled
case -c	Keyboard use cupola is set.
case -C	catc mode is set.
case -d	Debugging is turned on.
case -D	Debugging for static vehicles is turned on.
case -e	The ethernet is closed.
case -E	The exercise ID is set.
case -F	cfail debug is on.
case -g	The CIG isn't using graphics.
case -G	Guise override is set on.
case -h	Print help
case -?	Print help
case -k	The keyboard is in use.
case -l	Not used.
case -m	Status printing is disabled.
case -n	Verbose mode is used.
case -N	Night vision is enabled.
case -o	Printing of overruns is enabled.
case -p	The simulator is started in stand alone mode. The simulator acts as if it received an activation packet from the MCC. This segment of code is similar to that used by the MCC for activating a simulator.
case -P	Proximity list debugging is on.
case -s	Sound isn't used.
case -S	The network device is set.
case -t	Use the database override named.
case -T	The DED names are set.
case -v	Terrain verbos3 mode is on.
case -1	CIG1 is set.
case -2	CIG2 is set.
case -z	The vehicle size scale is changed.

Parameters		
Parameter	Type	Where Typedef Declared
argc	int	Standard
argv	pointer to char	Standard

Internal Variables		
Internal Variable	Type	Where Typedef Declared
i	int	Standard
initial heading	float	Standard
status	pointer to SIMNET M1 Status	stat_m1.h
gp	pointer to GroundVehicleSubsystems	status.h
unit	pointer to OrganizationalUnit	basic.h
status_bits	unsigned int	Standard
Calls		
Function	Where Described	
enter gracefully	Section 2.5.1.1.1	
network set exercise id	Section 2.1.1.3.1.49.1	
main read pars file	Section 2.5.1.2.1	
set request receive size	Section 2.1.2.2.1.29.1	
set request send size	Section 2.1.2.2.1.30.1	
set assymetric on	Section 2.1.2.2.2.112.1	
keyboard use cupola	Section 2.1.5.2	
set catc mode	Section 2.2.6.4.3	
use static debug	Section 2.1.2.2.2.116.1	
network dont really open_u p ethernet	Section 2.1.1.3.2.14.1	
network set exercise id	Section 2.1.1.3.1.49.1	
cfail debug on	Section 2.5.4.2.1	
cig not using graphics	Section 2.1.2.2.1.5.1	
print help	Section 2.2.1.1.1	
keyboard really use	Section 2.1.6.2	
disable status printing		
v pkt verbose mode	Section 2.1.1.3.1.66.1	
vision set otw night vision	Section 2.2.6.4.3	
get default db name	Section 2.5.1.2.15	
get default db version	Section 2.5.1.2.16	
rva turn debug on	Section 2.5.12.2.1	
sound dont use	Section 2.1.3.2	
network set network device	Section 2.1.1.3.2.12.4	
cig_use_database_override_ named	Section 2.1.2.2.1.16.1	
isalpha		
set ded name	Section 2.1.2.2.1.8.2	
terrain verbose mode on	Section 2.5.11.9.1	
set cig dev	Section 2.1.2.2.1.26.1	
set cig mask	Section 2.1.2.2.2.114.1	
sim state startup	Section 2.5.1.1.5	
simulation state machine	Section 2.5.1.1.13	

Table 2.2-8: main Information.

2.2.1.1.10 reconstitute_vehicle

This routine reconstitutes a vehicle, sending an activate request to the MCC.

Calls	
Function	Where Described
process activate request	2.1.1.3.2.1.1
network get exercise id	2.1.1.3.1.16.1

Table 2.2-9: reconstitute_vehicle Information.

2.2.2 M1 Controls/Switchology

A large portion of the code generated for specific vehicles exists in this section. Since every vehicle has its own complement of differing controls and indicators, these files exist in vehicle specific code. The structure of this CSC is shown in Figure 2.2-4. The three third level CSC's which comprise this second level CSC are as follows:

Low Level Control Handling
Finite State Machines
Specialized Output Devices

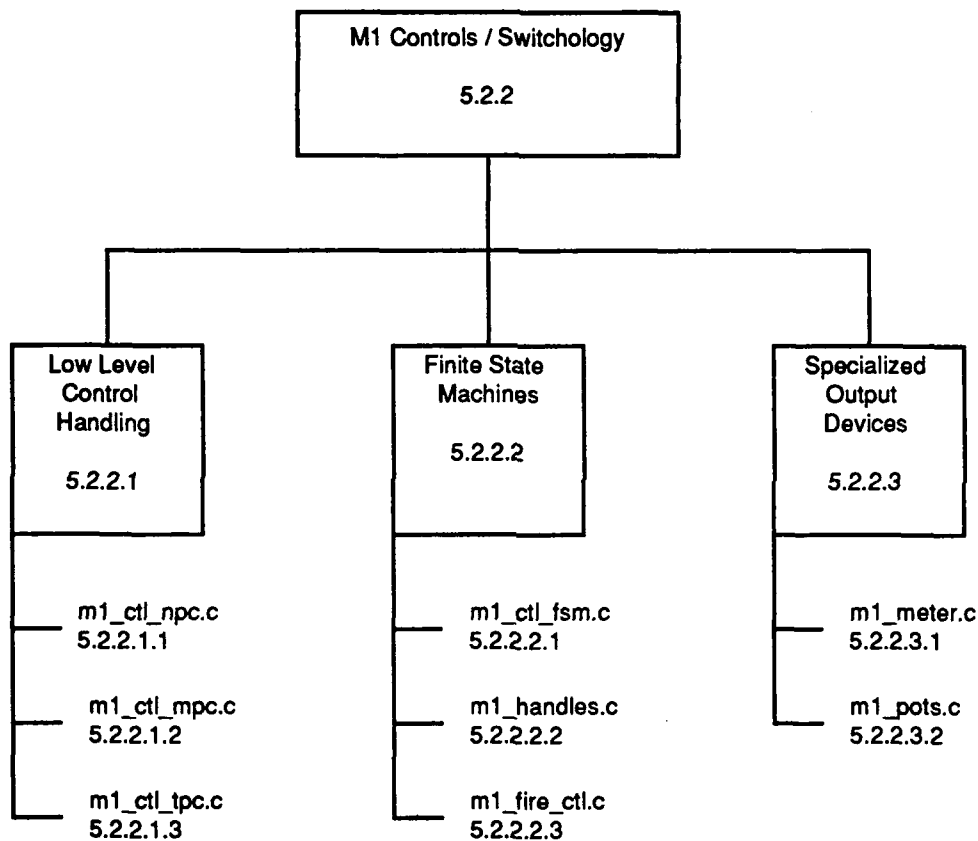


Figure 2.2-3: M1 Controls / Switchology CSC structure.

2.2.2.1 Low Level Control Handling

Since the simulation is interested in transitions of a control, the past state of a control must be maintained and updated each tick of the simulation. Therefore, a state variable is defined for each control in a vehicle simulation. The controls are grouped into separate files based on the "power state". The "power state" of a control is defined as the power requirements to make the control active. The following controls are examples:

To increase efficiency, the past state of a control is only maintained when a control is active. The last three letters of the files listed below are the initials of the power states:

- mpc: master power controls
- npc: no power controls
- tpc: turret power controls
- tnp: turret no power
- hnp: hull no power
- tdc: turret drive controls

The CSU's that are associated with this CSC are as follows:

- ml_ctl_npc.c
- ml_ctl_mpc.c
- ml_ctl_tpc.c
- ml_pots.c

2.3.2.1.1 m1_ctl_npc.c

(/simnet/release/src/vehicle/m1/src/m1_ctl_npc.c [m1_ctl_npc.c])

This CSU provides the no power controls interface for the M1 simulator. No power is required to make the control active. This module uses these include files:

"stdio.h"	"sim_types.h"
"sim_dfns.h"	"sim_macros.h"
"mass_std.h"	"dgi_stdg.h"
"sim_cig_if.h"	
"libcig.h"	"dtad.h"
"libidc.h"	"libidc_dfn.h"
"libmem.h"	"libmem_dfn.h"
"libnetwork.h"	"status.h"
"timers.h"	"timers_dfn.h"
"m1_ammo.h"	"m1_ammo_df.h"
"m1_ammo_mx.h"	"m1_ammo_pn.h"
"m1_cntrl.h"	"m1_ctl_df.h"
"m1_cupola.h"	"m1_driv_pn.h"
"m1_dtrain.h"	"m1_gunn_mx.h"
"m1_comm_mx.h"	"m1_load_mx.h"
"m1_hydrsys.h"	"m1_idc.h"
"m1_pots.h"	"m1_repair.h"
"m1_status.h"	"m1_thermal.h"
"m1_tmrs_df.h"	"m1_tracks.h"
"m1_turr_pn.h"	"m1_turret.h"
"m1_vision.h"	"mun_type.h"

This file defines the following variables:

Variables		
Variables	Type	Where Typedef Declared
apds translations	int array	Standard
heat translations	int array	Standard
select translations	int array	Standard
turret ref translations	int array	Standard
real service brake val	REAL	sim_types.h
real comm weap val	REAL	sim_types.h
real load peri val	REAL	sim_types.h
hex service brake val	int	Standard
hex comm weap val	int	Standard
hex load peri val	int	Standard
gps mag val	char	Standard
breech ready	char	Standard
ejection guard val	char	Standard
ammo transfer status	char	Standard
grid azimuth status	char	Standard
fuel flash count	int	Standard
fuel flash status	char	Standard
odometer timer number	int	Standard
cupola up down val	char	Standard
binoculars on off val	char	Standard
lpscope up down val	char	Standard
thermal shutter val	char	Standard

Table 2.2-10: m1_ctl_npc.c variables.

This file defines the routines listed in the tables below.

Initialization Routines:

The routines listed in the following table initialize the service brake, the gunner's primary sight magnification, the ejection guard, the ammunition transfer system, the commander's weapon station, the loader's periscope, the cupola, the binoculars, and the thermal shutter.

Name	Type of Routine
controls npc init ()	void
controls service brake init ()	void
controls mag init ()	void
controls ejection guard init ()	void
controls ammo transfer init ()	void
controls commander weapon station init ()	void
controls loader periscope init ()	void
controls cupola up down init ()	void
controls binoculars on off init ()	void
controls lpscope up down init ()	void
controls thermal shutter init ()	void

Table 2.2-11: Initialization routines.

Status Indicator Check Routines:

These routines are called on a tick by tick basis to check the status of the parking brake, service brake, ejection guard, breech, ammunition transfer system, knee switch, commander's weapon station, the loader's periscope, the fuel controls, odometer, cupola, binoculars, and thermal shutter.

Name	Type of Routine
controls parking brake check ()	void
controls service brake check ()	void
controls mag check ()	void
controls ejection guard check ()	void
controls breech check ()	void
controls breech unload check ()	void
controls breech ready check ()	void
controls ammo transfer check ()	void
controls knee switch check ()	void
controls commander weapon station check()	void
controls loader periscope check ()	void
controls grid azimuth check ()	void
controls fuel flash check ()	void
controls odometer check ()	void
controls cupola up down check ()	void
controls binoculars on off check ()	void
controls lpscope up down check ()	void

Table 2.2-12: Status Indication Check Routines.

Exit Routines:

These routines cause an orderly exit from the service brake, odometer, cupola, binoculars, and loader's periscope.

Name	Type of Routine
controls no power routines ()	void
controls service brake exit ()	void
controls odometer exit ()	void
controls cupola up down exit ()	void
controls binoculars on off exit ()	void
controls lpscope up down exit ()	void
controls no power off ()	void

Table 2.2-13: Exit Routines.

Munitions Management Controls Routines:

These routines are called internally to interface with the switches and indicators on the Ammo Redistribute panel, the Loader's Panel, the ammo tube, the breech, the Ammunition Ready Rack, and the fuel redistribution system.

Name	Type of Routine
controls transfer semi heat ()	void
controls transfer semi apds ()	void
controls transfer hull heat ()	void
controls transfer hull apds ()	void
controls transfer no transfer ()	void
controls transfer redist send ()	void
controls transfer redist recv ()	void
controls ejection guard armed ()	void
controls ejection guard safe ()	void
controls ammo tube check ()	void
controls show round ()	void
controls unshow round ()	void
controls resupply empty ()	void
controls resupply flash ()	void
controls resupply unflash ()	void
controls resupply restore ()	void
controls show breech ()	void
controls start fuel flashing ()	void
controls stop fuel flashing ()	void
controls restore ammo ()	void
controls fuel flash ()	void
controls fuel unflash ()	void
controls fuel restore ()	void

Table 2.2-14: Munitions Management Routines.

Miscellaneous Routines:

These routines are called internally for control of the parking brake, control of the turret reference indicator, and global access of the gunner's primary sight value.

Name	Type of Routine
get non thermal mag ()	char
controls set parking brake ()	void
controls release parking brake ()	void
controls turret ref ind ()	void

Table 2.2-15: Miscellaneous Routines.

2.3.2.1.2 m1_ctl_mpc.c

(/simnet/release/src/vehicle/m1/src/m1_ctl_mpc.c [m1_ctl_mpc.c])

This CSU provides the master power controls interface for the M1 simulator. These controls require master control power for activation. This module uses these include files:

"stdio.h"	"sim_types.h"
"sim_dfns.h"	"sim_macros.h"
"timers_dfn.h"	"timers.h"
"libidc.h"	"libidc_dfn.h"
"libmem.h"	"libmem_dfn.h"
"libnetwork.h"	
"m1_cntrl.h"	"m1_ctl_df.h"
"m1_electsys.h"	"m1_engine.h"
"m1_driv_mx.h"	"m1_driv_pn.h"
"m1_dtrain.h"	"m1_fuelsys.h"
"m1_hydrsys.h"	"m1_meter.h"
"m1_pots.h"	"m1_turr_pn.h"

This file defines the following variables.

Variables		
Variables	Type	Where Typedef Declared
real steer bar val	REAL	sim_types.h
real throttle val	REAL	sim_types.h
hex steer bar val	int	Standard
hex throttle val	int	Standard
driver panel test val	char	Standard
transmission val	char	Standard
tank select val	char	Standard
tactical idle val	char	Standard
push to start val	char	Standard
engine lamp timer number	int	Standard
engine abort timer number	int	Standard
radio failure status	int	Standard

Table 2.2-16: m1_ctl_mpc.c variables.

This file defines the routines listed in the tables below.

Initialization Routines:

The routines listed in the following table initialize the controls requiring master power: steer bar, throttle, transmission, fuel tank selector, engine start switch, driver panel lights, and tactical idle switch.

Name	Type of Routine
controls_mpc_init ()	void
controls_master_power_on ()	void
controls_steer_bar_init ()	void
controls_throttle_init ()	void
controls_transmission_init ()	void
controls_tank_select_init ()	void
controls_engine_start_init ()	void
controls_driver_panel_light_init ()	void
controls_tactical_idle_init ()	void

Table 2.2-17: Initialization routines.

Status Indication Routines:

These routines are called on a tick by tick basis to check the status of the steer bar, throttle, transmission, fuel tank selector switch, driver panel lights, engine start switch, engine shutoff switch, engine lamp, engine tactical idle switch, and master caution lamps.

Name	Type of Routine
controls_master_power_routines ()	void
controls_steer_bar_check ()	void
controls_throttle_check ()	void
controls_transmission_check ()	void
controls_tank_select_check ()	void
controls_engine_start_check ()	void
controls_engine_shutoff_check ()	void
controls_engine_lamp_check ()	void
controls_engine_abort_check ()	void
controls_tactical_idle_check ()	void
controls_caution_reset_check ()	void
controls_driver_panel_light_check ()	void

Table 2.2-18: Status Indication Routines.

Exit Routines:

These routines cause an orderly exit from the routines listed above.

Name and Description	Type of Routine
controls master power off ()	void
controls steer bar exit ()	void
controls throttle exit ()	void
controls engine lamp exit ()	void
controls engine abort exit ()	void
controls driver panel light exit ()	void

Table 2.2-19: Exit Routines.

Miscellaneous Routines:

These routines are used internally by the above routines.

Name and Description	Type of Routine
controls driver panel light on ()	void
controls driver panel light restore ()	void
controls master power edges clear ()	void
controls driver panel status ()	int
controls engine started ()	void
controls engine spooling down ()	void
controls engine abort ()	void
controls engine reset abort ()	void
controls engine overspeed ()	void
controls engine overspeed normal ()	void
controls low fuel on ()	void
controls low fuel off ()	void
controls low charge on ()	void
controls low charge off ()	void
controls engine oil level low ()	void
controls engine oil level normal ()	void
controls transmission oil level low ()	void
controls transmission oil level normal ()	void
controls engine oil filter clogged ()	void
controls engine oil filter normal ()	void
controls transmission oil filter clogged ()	void
controls transmission oil filter normal ()	void
controls engine fuel filter clogged ()	void
controls engine fuel filter normal ()	void
controls right pump inoperative on ()	void
controls right pump inoperative off ()	void
controls left pump inoperative on ()	void
controls left pump inoperative off ()	void
controls engine oil temperature high ()	void
controls engine oil temperature normal ()	void
controls engine oil pressure low ()	void
controls engine oil pressure normal ()	void
controls transmission oil temperature high ()	void
controls_transmission_oil_temperature_normal ()	void
controls transmission oil pressure low ()	void
controls transmission oil pressure normal ()	void
controls caution lamp off check ()	void
controls warning lamp off check ()	void
controls kill radio ()	void

Table 2.2-20: Miscellaneous Routines.

2.3.2.1.2 m1_ctl_tpc.c

(/simnet/release/src/vehicle/m1/src/m1_ctl_tpc.c [m1_ctl_tpc.c])

This CSU provides the turret power controls interface for the M1 simulator. These controls require turret power for activation. This module uses these include files:

"stdio.h"	"sim_types.h"
"sim_dfns.h"	"sim_macros.h"
"timers_dfn.h"	"timers.h"
"libidc.h"	"libidc_dfn.h"
"libmem.h"	"libmem_dfn.h"
"libnetwork.h"	"m1_bcs.h"
"m1_cntrl.h"	"m1_ctl_df.h"
"m1_ammo_df.h"	"m1_ammo.h"
"m1_firectl.h"	"m1_weapons.h"
"m1_handles.h"	"m1_laser.h"
"m1_pots.h"	"m1_turret.h"
"m1_comm_mx.h"	"m1_gunn_mx.h"
"m1_load_mx.h"	"m1_turr_pn.h"
"m1_ammo_pn.h"	"m1_thermal.h"

This file defines the following variables.

Variables		
Variables	Type	Where Typedef Declared
real gunn elev val	REAL	sim types.h
real comm elev val	REAL	sim types.h
real gunn trav val	REAL	sim types.h
real comm trav val	REAL	sim types.h
hex gunn elev val	int	sim types.h
hex comm elev val	int	sim types.h
hex gunn trav val	int	sim types.h
hex comm trav val	int	sim types.h
commander add drop val	char	sim types.h
commander panel test val	char	sim types.h
ammo select val	char	sim types.h
fire control val	char	sim types.h
gun drive val	char	sim types.h
gun select val	char	sim types.h
gunner palm switch val	char	sim types.h
commander palm switch val	char	sim types.h
laser select val	char	sim types.h
gunner palm switch val	char	sim types.h
commander palm switch val	char	sim types.h
laser select val	char	sim types.h
gunner laser button val	char	sim types.h
commander laser button val	char	sim types.h
thermal mode val	char	sim types.h
thermal polarity val	char	sim types.h
thermal magnitude val	char	sim types.h

Table 2.2-21: m1_ctl_tpc.c variables.

This file defines the routines listed in the tables below.

Initialization Routines:

The routines listed in the following table initialize the gunner's and commander's palm switches, the gunner's and commander's gun elevation handles, the gunner's and commander's gun traverse handles, the gunner's and commander's laser buttons, the bcs manual range drop button, the commander's panel lights, the fire control gun drive control, the gun selector, the ammo selector, the thermal mode control, the thermal polarity control, and the thermal magnitude control.

Name and Description	Type of Routine
controls palm switch init ()	void
controls gun elevation init ()	void
controls gun traverse init ()	void
controls laser select init ()	void
controls add drop init ()	void
controls firectl gundrive init ()	void
controls gun select init ()	void
controls ammo select init ()	void
controls commander panel light init ()	void
controls thermal mode init ()	void
controls thermal polarity init ()	void
controls thermal magnitude init ()	void

Table 2.2-22: Initialization routines.

Miscellaneous Status Indicator Routines:

These routines are called on a tick by tick basis to check the status of the controls initialized in the table above.

Name and Description	Type of Routine
controls palm switch check ()	void
controls gun elevation check ()	void
controls gun traverse check ()	void
controls laser fired check ()	void
controls laser select check ()	void
controls trigger check ()	void
controls battlesight check ()	void
controls add drop check ()	void
controls firectl gundrive check ()	void
controls gun select check ()	void
controls ammo select check ()	void
controls commander panel light check ()	void
controls thermal mode check ()	void
controls thermal polarity check ()	void
controls thermal magnitude check ()	void

Table 2.2-23: Check Routines.

Exit Routines:

These routines cause an orderly exit from the routines listed above.

Name and Description	Type of Routine
controls palm switch exit ()	void
controls gun elevation exit ()	void
controls gun traverse exit ()	void
controls add drop exit ()	void
controls commander panel light exit ()	void
controls thermal mode exit ()	void
controls thermal polarity exit ()	void
controls thermal magnitude exit ()	void

Table 2.2-24: Exit Routines.

Miscellaneous Routines:

These routines are used internally by the above routines.

Name and Description	Type of Routine
controls turret power off ()	void
controls gun fired ()	void
controls laser malfunction set ()	void
controls laser malfunction reset ()	void
controls thermal ready light on ()	void
controls thermal ready light off ()	void
controls firectrl gundrive fsm ()	void
controls gun select fsm ()	void
controls commander panel light on ()	void
controls commander panel light restore ()	void
controls commander fake light on ()	void
controls commander fake light restore ()	void

Table 2.2-25: Miscellaneous Routines.

2.2.2.2 Finite State Machines

In general, finite state machines describe the appropriate actions to take when a control changes value. The following are examples of the type of functions that are contained in the files listed below:

- ii) When the driver of an m2 tank presses the accelerator, the simulation must first discern the states of the engine, transmission, emergency brake, etc. before the action's effect can be determined
- iii) When the gunner on an m1 tank pulls the trigger, the simulation must first check the states of the gunners palm grip, commanders palm grip, the weapon loaded, etc. before determining the necessary actions resulting from the trigger pull.

The CSU's associated with this CSC are as follows:

m1_ctl_fsm.c
m1_handles.c
m1_firectl.c

2.2.2.2.1 m1_ctl_fsm.c (./simnet/release/src/vehicle/m1/src/m1_ctl_fsm.c [m1_ctl_fsm.c])

The CSU m1_ctl_fsm.c maintains the "power state" which is defined in section 2.2.2.1. This CSU provides the finite power state controls interface for the M1 simulator. This module uses these include files:

"stdio.h"	"sim_types.h"
"sim_dfns.h"	"libidc.h"
"libidc_dfn.h"	"libmem.h"
"libmem_dfn.h"	"libnetwork.h"
"m1_ammo.h"	"m1_cntrl.h"
"m1_ctl_df.h"	"m1_electsys.h"
"m1_engine.h"	"m1_driv_pn.h"
"m1_turr_pn.h"	"m1_ammo_pn.h"

This file defines the following variables:

Variables		
Variables	Type	Where Typedef Declared
controls status	int	Standard
controls electsys val	int	Standard
controls electsys off edge	int	Standard
controls failure val	int	Standard
controls failure edge	int	Standard

Table 2.2-27: m1_ctl_fsm.c variables.

Simulation Routines:

The `controls_simul()` routine contains the fine state machine simulation, called on a tick by tick basis. The other routines handle the transitions between states in the master power, no power, and turret power conditions. If a no power state is detected, then the controls are set to specified initial values which are usually defined as an OFF state.

Name and Description	Type of Routine
<code>controls_no_power_next_state()</code>	void
<code>controls_master_power_next_state()</code>	void
<code>controls_turret_power_next_state()</code>	void
<code>controls_simul()</code>	void

Table 2.2-28: Simulation Routines.

Initialization Routines:

These routines initialize the finite state machine, the edges, and the IDC outputs.

Name and Description	Type of Routine
<code>controls_fsm_init()</code>	void
<code>controls_edge_init()</code>	void
<code>controls_lamp_init()</code>	void

Table 2.2-29: Initialization routines.

Miscellaneous Routines:

The following routines used internally by the above routines.

Name and Description	Type of Routine
<code>controls_power_status()</code>	void
<code>controls_electsys_dead()</code>	void
<code>controls_electsys_reborn()</code>	void
<code>controls_electsys_status()</code>	int
<code>controls_break_controls()</code>	void
<code>controls_restore_controls()</code>	void
<code>controls_failure_status()</code>	int
<code>controls_other_edges_clear()</code>	void

Table 2.2-30: Miscellaneous Routines.

2.2.2.2.2 m1_handles.c

(./simnet/release/src/vehicle/m1/src/m1_handles.c [m1_handles.c])

The CSU m1_handles.c contains state machines that keep track of the gunners and commanders palm grips, and the associated control of the respective turrets.

Includes:

```
"sim_types.h"
"sim_dfns.h"
"libfail.h"
"failure.h"
"m1_bcs.h"
"m1_weapons.h"
"m1_laser.h"
"m1_turret.h"
```

int variable declarations:

gunner_control_status	-- maintenance status: normally WORKING
cmdr_control_status	
gunner_control_engaged	-- booleans which tell whether palm switches are engaged
cmdr_control_engaged	
gunner_trigger_pushed	-- either TRUE or FALSE
cmdr_trigger_pushed	-- These are both edge-triggered rather than level-triggered, so they are cleared immediately after use.
gunner_laser_pushed	-- either CLEAR, PUSHED, or USED. These are both level-triggered. A stock M1 treats them as edges.
cmdr_laser_pushed	
use_dazzler_laser	

REAL variable declarations:

gunner_traverse	-- normalized values from -1.0 to +1.0 which are received directly from controls and passed to the turret if the appropriate palm switch is engaged
cmdr_traverse	
gunner_elevate	
cmdr_elevate	

Defines:

<u>Symbol</u>	<u>Value</u>
CLEAR	0
PUSHED	1
USED	2

2.2.2.2.2.1 handles_simul

This routine simulates the commander's controls on a tick by tick basis. It checks to see if handles are working and which handles are engaged.

Calls	
Function	Where Described
laser lase	Section 2.2.3.2.15
turret handles values	Section 2.2.6.1.1.6
weapons fire main gun	Section 2.2.3.3.6

Table 2.2-31: handles_simul Information.

2.2.2.2.2.2 handles_gunner_control_fixed

This routine repairs the gunner's control.

2.2.2.2.2.3 handles_gunner_control_broken

This routine causes the gunner's control to fail.

2.2.2.2.2.4 handles_commander_control_fixed

This routine repairs the commander's control.

2.2.2.2.2.5 handles_commander_control_broken

This routine causes the commander's control to fail.

2.2.2.2.2.6 handles_gunner_palm_on

This routine engages the gunner's control.

2.2.2.2.2.7 handles_gunner_palm_off

This routine disengages the gunner's palm switch. It is called by the controls module when the gunner's palm switch is released, or when turret power is turned off while the gunner's palm switch is engaged.

2.2.2.2.2.8 handles_commander_palm_on

This routine engages the commander's palm switch.

2.2.2.2.2.9 handles_commander_palm_off

This routine disengages the commander's palm switch. It is called by the controls module when the commander's palm switch is released, or when turret power is turned off while the commander's palm switch is engaged.

2.2.2.2.2.10 handles_set_gunner_elevation

This routine sets the gunner's elevation rate to *elevation_rate*.

Parameters		
Parameter	Type	Where Typedef Declared
elevation_rate	register REAL	sim_types.h

Table 2.2-32: handles_set_gunner_elevation Information.

2.2.2.2.2.11 handles_set_commander_elevation

This routine sets the commander's elevation rate to *elevation_rate*.

Parameters		
Parameter	Type	Where Typedef Declared
elevation_rate	register REAL	sim_types.h

Table 2.2-33: handles_set_commander_elevation Information.

2.2.2.2.2.12 handles_set_gunner_traverse

This routine sets the gunner's traverse rate to *traverse_rate*.

Parameters		
Parameter	Type	Where Typedef Declared
traverse_rate	register REAL	sim_types.h

Table 2.2-34: handles_set_gunner_traverse Information.

2.2.2.2.2.13 handles_set_commander_traverse

This routine sets the commander's traverse rate to *traverse_rate*.

Parameters		
Parameter	Type	Where Typedef Declared
traverse_rate	register REAL	sim_types.h

Table 2.2-35: handles_set_commander_traverse Information.

2.2.2.2.2.14 handles_gunner_laser_button_released

This routine is not used in the version 6.6 release.

2.2.2.2.2.15 handles_gunner_laser_button_depressed

This routine sets gunner_laser_pushed equal to PUSHED when the gunner's laser button is depressed.

2.2.2.2.2.16 handles_commander_laser_button_released

This routine sets cmdr_laser_pushed equal to CLEAR when the commander's laser button is released.

2.2.2.2.2.17 handles_commander_laser_button_depressed

This routine sets cmdr_laser_pushed equal to PUSHED when the commander's laser button is depressed.

2.2.2.2.2.18 handles_gunner_trigger_depressed

This routine sets gunner_trigger_pushed equal to TRUE when the gunner's trigger is depressed.

2.2.2.2.2.19 handles_commander_trigger_depressed

This routine sets cmdr_trigger_pushed equal to TRUE when the commander's trigger is depressed.

2.2.2.2.2.20 handles_gunner_trigger_released

This routine is not used in the version 6.6 release.

2.2.2.2.2.21 handles_commander_trigger_released

This routine is not used in the version 6.6 release.

2.2.2.2.2.22 handles_init

This routine initializes the handles.

Calls	
Function	Where Described
fail init failure	Section 2.5.4.11.2

Table 2.2-36: handles_init Information.

2.2.2.2.3 m1_firectl.c

(./simnet/release/src/vehicle/m1/src/m1_firectl.c [m1_firectl.c])

The CSU m1_firectl.c computes the states important to firing weapons.

Includes:

```
"sim_types.h"
"sim_dfns.h"
"m1_gunn_mx.h"
"m1_load_mx.h"
"m1_ctl_df.h"
"m1_cntrl.h"
"m1_firectl.h"
"m1_laser.h"
"m1_ammo.h"
```

int variable declarations:

```
gun_selected
fire_control_mode
gun_drive_val
```

BOOLEAN variable declarations:

```
malfunction_val
```

2.2.2.2.3.1 firectl_init

This routine initializes the fire control state variables gun_selected, fire_control_mode, gun_drive_val, and malfunction_val.

2.2.2.2.3.2 firectl_malfunction

This routine returns the fire control malfunction state.

Return Values		
Return Value	Type	Meaning
malfunction_val	BOOLEAN	The fire control malfunction state

Table 2.2-37: firectl_malfunction Information.

2.2.2.2.3.3 firectl_laser_malfunction_set

This routine sets the fire control malfunction state.

Calls	
Function	Where Described
controls_laser_malfunction_set	Section 2.2.2

Table 2.2-38: firectl_laser_malfunction_set Information.

2.2.2.2.3.4 firectl_laser_malfunction_reset

This routine clears the fire control malfunction state.

Calls	
Function	Where Described
controls_laser_malfunction_reset	Section 2.2.2

Table 2.2-39: firectl_laser_malfunction_reset Information.

2.2.2.2.3.5 firectl_ready_to_fire

This routine returns the fire control ready status. The fire control system is ready if the ammunition module is ready to fire, the turret power is on, the main gun is selected, the fire control system is not in manual mode, and the gun drive state is powered.

Return Values		
Return Value	Type	Meaning
(ammo_ready_to_fire()) && (controls_power_status() == CONTROLS_STATE_TURRET_POWER_ON) && (gun_selected == GN_MAIN_VAL) && (fire_control_mode != GN_MANUAL_VAL) && (gun_drive_val == LD_POWERED_VAL)	BOOLEAN	If TRUE, the fire control system is ready; If FALSE, the fire control system is not ready.
Calls		
Function	Where Described	
ammo ready to fire	Section 2.2.5.1.32	
controls power status	Section 2.2.2	

Table 2.2-40: firectl_ready_to_fire Information.

2.2.2.2.3.6 firectl_gun_select_main

This routine sets the gun select state to main.

2.2.2.2.3.7 firectl_gun_select_safe

This routine sets the gun select state to safe.

2.2.2.2.3.8 firectl_gun_select_coax

This routine sets the gun select state to coaxial.

2.2.2.2.3.9 firectl_gun_select_status

This routine returns the gun select state.

Return Values		
Return Value	Type	Meaning
gun_selected	int	The gun select state

Table 2.2-41: firectl_gun_select_status Information.

2.2.2.2.3.10 firectl_fire_control_normal

This routine sets the fire control mode to normal.

2.2.2.2.3.11 firectl_fire_control_emergency

This routine sets the fire control mode to emergency.

2.2.2.2.3.12 firectl_fire_control_manual

This routine sets the fire control mode to manual.

2.2.2.2.3.13 firectl_fire_control_status

This routine returns the fire control mode.

Return Values		
Return Value	Type	Meaning
fire_control_mode	int	The fire control mode

Table 2.2-42: firectl_fire_control_status Information.

2.2.2.2.3.14 firectl_gun_drive_manual

This routine sets the gun drive state to manual.

2.2.2.2.3.15 firectl_gun_drive_powered

This routine sets the gun drive state to powered.

2.2.2.2.3.16 firectl_gun_drive_uncpl

This routine sets the gun drive state to uncoupled.

2.2.2.2.3.17 firectl_gun_drive_status

This routine returns the gun drive state.

Return Values		
Return Value	Type	Meaning
gun_drive_val	int	The gun drive state

Table 2.2-43: firectl_gun_drive_status Information.

2.2.2.3 Specialized Output Devices

As stated in section 2.1, the output devices for a simulation are often vehicle specific. The files listed below contain routine which compute values to be displayed to the operator. The new values are sent to the low level control routines (see 2.1.4). This functionality is realized by two CSU's, m1_meter.c and m1_pots.c.

2.2.2.3.1 m1_meter.c

(./simnet/release/src/vehicle/m1/src/m1_meter.c [m1_meter.c])

This CSU provides routines which simulate meters on the M1.

Includes:

```
"sim_types.h"
"sim_dfns.h"
"m1_mtr_df.h"
"m1_ctl_df.h"
"m1_fuel_df.h"
"m1_driv_pn.h"
"m1_meter.h"
"m1_cntrl.h"
```

char declarations:

```
speed_set_val
tach_set_val
fuel_set_val
volt_set_val
```

REAL variable declarations and initialization:

```
fuel_full = REAR_FUEL_FULL
```

2.2.2.3.1.1 meter_init

This routine initializes the four meter outputs: speedometer, tachometer, fuel guage, and voltmeter.

2.2.2.3.1.2 meter_speed_set

This routine sets the reading on the speedometer to *val*, the input value.

Parameters		
Parameter	Type	Where Typedef Declared
val	REAL	sim_types.h
Internal Variables		
Internal Variable	Type	Where Typedef Declared
temp1	char	Standard
temp2	int	Standard
Calls		
Function	Where Described	
controls power status	Section 2.2.2	
controls electsys status	Section 2.2.2	
controls failure status	Section 2.2.2	
idc output set	Section 2.1.4.1.1	

Table 2.2-44: meter_speed_set Information.

2.2.2.3.1.3 meter_tach_set

This routine sets the value on the tachometer based on the value of *val*.

Parameters		
Parameter	Type	Where Typedef Declared
val	REAL	sim_types.h
Internal Variables		
Internal Variable	Type	Where Typedef Declared
temp1	char	Standard
temp2	int	Standard
Calls		
Function	Where Described	
controls power status	Section 2.2.2	
controls electsys status	Section 2.2.2	
controls failure status	Section 2.2.2	
idc output set	Section 2.1.4.1.1	

Table 2.2-45: meter_tach_set Information.

2.2.2.3.1.4 meter_fuel_set

This routine sets the output on the fuel guage based on the value of *val*.

Parameters		
Parameter	Type	Where Typedef Declared
val	REAL	sim_types.h
Internal Variables		
Internal Variable	Type	Where Typedef Declared
temp1	char	Standard
temp2	int	Standard
Calls		
Function	Where Described	
controls power status	Section 2.2.2	
controls electsys status	Section 2.2.2	
controls failure status	Section 2.2.2	
idc output set	Section 2.1.4.1	

Table 2.2-46: meter_fuel_set Information.

2.2.2.3.1.5 meter_volt_set

This routine sets the value on the voltmeter based on the value of *val*.

Parameters		
Parameter	Type	Where Typedef Declared
val	REAL	sim_types.h
Internal Variables		
Internal Variable	Type	Where Typedef Declared
temp1	char	Standard
temp2	int	Standard
Calls		
Function	Where Described	
controls power status	Section 2.2.2	
controls electsys status	Section 2.2.2	
controls failure status	Section 2.2.2	
idc output set	Section 2.1.4.1	

Table 2.2-47: meter_volt_set Information.

2.2.2.3.1.6 meter_select_front_left_tank

This routine selects the left front fuel tank.

2.2.2.3.1.7 meter_select_front_right_tank

This routine selects the right front fuel tank.

2.2.2.3.1.8 meter_select_rear_tank

This routine selects the left front fuel tank.

2.2.2.3.2 m1_pots.c

(./simnet/release/src/vehicle/m1/src/m1_pots.c [m1_pots.c])

This file contains the procedures relating to the potentiometers, which translate hex potentiometer values between 00 and FF into real numbers and call the appropriate M1 subsystems with those real values.

Includes:

- "stdio.h"
- "sim_types.h"
- "sim_dfns.h"
- "sim_macros.h"
- "libidc_dfn.h"
- "libnetwork.h"
- "m1_pots_df.h"
- "m1_cali_df.h"
- "libpots.h"

The following functions are defined for the Butterfly:

- FOPEN
- FCLOSE
- FSCANF1
- FSCANF2

The following variables represent the calibrated hex values corresponding to the pot extremes, i.e., cm_tur_trav_l is the hex value of the commanders turret traverse handle when it is at the full left position. These values are used to convert hex pot values to real numbers.

The commander's turret traverse: left, right, centered:

- cm_tur_trav_l
- cm_tur_trav_r
- cm_tur_trav_c

The commander's turret elevate: depress, raise, centered:

- cm_tur_elev_d
- cm_tur_elev_r
- cm_tur_elev_c

The gunner's turret traverse: left, right, centered:

gn_tur_trav_l
gn_tur_trav_r
gn_tur_trav_c

The gunner's turret elevate: depress, raise, centered:

gn_tur_elev_d
gn_tur_elev_r
gn_tur_elev_c

The driver's steering bar: left, right, centered:

dr_steer_l
dr_steer_r
dr_steer_c

The driver's throttle: zero, full:

dr_throttle_z
dr_throttle_f

The driver's service brake: zero, full:

dr_s_brake_z
dr_s_brake_f

The commander's weapon station: left, right:

cm_cws_l
cm_cws_r

The loader's periscope: left, right:

ld_peri_l
ld_peri_r

2.2.2.3.2.1 pots_init

This routine initializes the potentiometers. The calibration file is opened, and the potentiometer variables are assigned values.

Internal Variables		
Internal Variable	Type	Where Typedef Declared
pots	pointer to FILE	
line	int	Standard
pots_error	int	Standard
Calls		
Function	Where Described	
pots_check_three	Section 2.1.4.1.2.6.1	
pots_check_two	Section 2.1.4.1.2.1.1	

Table 2.2-48: pots_init Information.

2.2.2.3.2.2 pots_comm_trav_real

Given the potentiometer value *pot*, this routine returns a scaled real value between -1.0 and +1.0 for the commander's turret traverse control.

Parameters		
Parameter	Type	Where Typedef Declared
pot	int	Standard
Return Values		
Return Value	Type	Meaning
pots_scale_lcr (pot, cm_tur_trav_l, cm_tur_trav_c, cm_tur_trav_r, COMM_TRAV_TOLERANCE)	REAL	The scaled value of the commander's turret traverse control
Calls		
Function	Where Described	
pots_scale_lcr	Section 2.1.4.1.2.2.1	

Table 2.2-49: pots_comm_trav_real Information.

2.2.2.3.2.3 pots_comm_elev_real

Given the potentiometer value *pot*, this routine returns a scaled real value between -1.0 and +1.0 for the commander's elevation control.

Parameters		
Parameter	Type	Where Typedef Declared
pot	int	Standard
Return Values		
Return Value	Type	Meaning
pots_scale_lcr (pot, cm_tur_elev_r, cm_tur_elev_c, cm_tur_elev_d, COMM_ELEV_TOLERANCE)	REAL	The scaled value of the commander's elevation control
Calls		
Function	Where Described	
pots_scale_lcr	Section 2.1.4.1.2.2.1	

Table 2.2-50: pots_comm_elev_real Information.

2.2.2.3.2.4 pots_gunn_trav_real

Given the potentiometer value *pot*, this routine returns a scaled real value between -1.0 and +1.0 for the gunner's turret traverse control.

Parameters		
Parameter	Type	Where Typedef Declared
pot	int	Standard
Return Values		
Return Value	Type	Meaning
pots_scale_lcr (pot, gn_tur_trav_l, gn_tur_trav_c, gn_tur_trav_r, GUNN_TRAV_TOLERANCE)	REAL	The scaled value of the gunner's turret traverse control
Calls		
Function	Where Described	
pots_scale_lcr	Section 2.1.4.1.2.2.1	

Table 2.2-51: pots_gunn_trav_real Information.

2.2.2.3.2.5 pots_gunn_elev_real

Given the potentiometer value *pot*, this routine returns a scaled real value between -1.0 and +1.0 for the gunner's elevation control.

Parameters		
Parameter	Type	Where Typedef Declared
pot	int	Standard
Return Values		
Return Value	Type	Meaning
pots_scale_lcr (pot, gn_tur_elev_r, gn_tur_elev_c, gn_tur_elev_d, GUNN_ELEV_TOLERANCE))	REAL	The scaled value of the gunner's elevation control
Calls		
Function	Where Described	
pots_scale_lcr	Section 2.1.4.1.2.2.1	

Table 2.2-52: pots_gunn_elev_real Information.

2.2.2.3.2.6 pots_steer_bar_real

Given the potentiometer value *pot*, this routine returns a scaled real value between -1.0 and +1.0 for the driver's steering bar.

Parameters		
Parameter	Type	Where Typedef Declared
pot	int	Standard
Return Values		
Return Value	Type	Meaning
pots_scale_lcr (pot, dr_steer_l, dr_steer_c, dr_steer_r, STEER_TOLERANCE)	REAL	The scaled value of the driver's steering bar
Calls		
Function	Where Described	
pots_scale_lcr	Section 2.1.4.1.2.2.1	

Table 2.2-53: pots_steer_bar_real Information.

2.2.2.3.2.7 pots_throttle_real

Given the potentiometer value *pots*, this routine returns a scaled real value between 0.0 and +1.0 for the driver's throttle.

Parameters		
Parameter	Type	Where Typedef Declared
pot	int	Standard
Return Values		
Return Value	Type	Meaning
pots_scale_lr_pos (pot, dr_throttle_z, dr_throttle f)	REAL	The scaled value of the driver's throttle
Calls		
Function	Where Described	
pots_scale_lr_pos	Section 2.1.4.1.2.4.1	

Table 2.2-54: pots_throttle_real Information.

2.2.2.3.2.8 pots_service_brake_real

Given the potentiometer value *pot*, this routine returns a scaled real value between 0.0 and +1.0 for the driver's service brake.

Parameters		
Parameter	Type	Where Typedef Declared
pot	int	Standard
Return Values		
Return Value	Type	Meaning
pots_scale_lr_pos (pot, dr s brake z, dr s brake f)	REAL	The scaled value of the driver's service brake
Calls		
Function	Where Described	
pots_scale_lr_pos	Section 2.1.4.1.2.4.1	

Table 2.2-55: pots_service_brake_real Information.

2.2.2.3.2.9 pots_comm_weap_real

Given the potentiometer value *pot*, this routine returns a scaled real value between -1.0 and +1.0 for the commander's weapons control.

Parameters		
Parameter	Type	Where Typedef Declared
pot	int	Standard
Return Values		
Return Value	Type	Meaning
pots_scale_lr_both (pot, cm cws l, cm cws r)	REAL	The scaled value of the commander's weapons control
Calls		
Function	Where Described	
pots_scale_lr_pos	Section 2.1.4.1.2.4.1	

Table 2.2-56: pots_comm_weap_real Information.

2.2.2.3.2.10 pots_load_peri_real

Given the potentiometer value *pot*, this routine returns a scaled real value between -1.0 and +1.0 for the loader's periscope control.

Parameters		
Parameter	Type	Where Typedef Declared
pot	int	Standard
Return Values		
Return Value	Type	Meaning
pots_scale_lr_both (pot, id_peri_l, id_peri_r)	REAL	The scaled value of the loader's periscope control
Calls		
Function	Where Described	
pots_scale_lr_pos	Section 2.1.4.1.2.4.1	

Table 2.2-57: pots_load_peri_real Information.

2.2.3 M1 Weapons

The M1 fires unguided, or ballistic, rounds. The simulation must model the processes associated with the weapons systems which take place within the vehicle. It also must model the flyout of the round and the reaction to impacts.

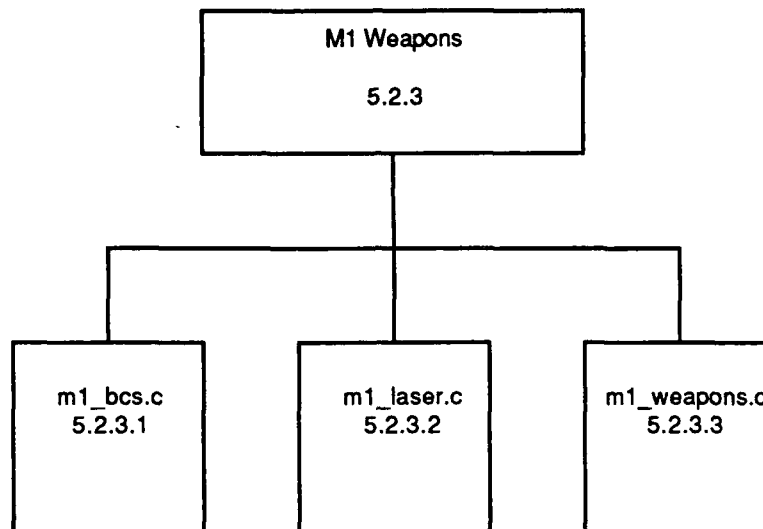


Figure 2.2-4: Structure of the M1 Weapons CSCI.

The M1 accounts for the ballistic trajectory of the rounds fired in ballistic computer systems. These systems take range to target and turret traverse rate information and knowledge about the round to be fired in order to calculate superelevation and lead angles.

The principle means of obtaining range information on the M1 is via the laser range finder. `m1_laser.c` models the laser system. It responds to commands from the controls and sends range information to the ballistics computer. The modes of operation of the system are modeled as are failure modes and multiple return conditions.

Calculation of the superelevation and lead angles take place in `m1_bcs.c`. The M1 uses range information provided by the laser or uses the range set by the user. The simulation of manual range setting is performed in `m1_bcs.c`. This routine tells the CIG the current range so the range can be displayed. Knowledge of the round to be fired is also maintained in `m1_bcs.c`.

Commands to fire a ballistic round are initiated in the controls code. There are a number of steps to be performed when a round is fired and these tasks are performed in `m1_weapons.c`. The availability of the round is checked by polling the munitions management code. If a round is available, the direction of the gun tube is determined. This is based on a combination of the gunner's line of sight, the superelevation and lead angle determined in the ballistics computer, and a random offset to model dispersion. The dispersion is found using routines in the math library. `Libball` is called to actually fire the round. If the round is successfully fired, the sound of firing is made, the ammunition is decremented and a message is put onto the net.

Some vehicle specific functions are associated with firing ballistic rounds performed in `m1_weapons.c`. The failure of the main gun on the M1 is also modeled in `m1_weapons.c`.

This CSC consists of three CSU's:

```
m1_bcs.c
m1_laser.c
m1_weapons.c
```

2.2.3.1 m1_bcs.c

(./simnet/release/src/vehicle/m1/src/m1_bcs.c [m1_bcs.c])

The M1's ballistics computer is simulated by this file.

Includes:

```
"stdio.h"
"math.h"
"sim_types.h"
"sim_dfns.h"
"sim_macros.h"
"basic.h"
"mun_type.h"
"timers.h"
"timers_dfn.h"
"libkin.h"
"libmatrix.h"
"libball.h"
"m1_bcs.h"
"m1_tracks.h"
"m1_turret.h"
```

Defines:

<u>Symbol</u>	<u>Value</u>	
BCS_DEBUG	FALSE	
BCS_NOT_CHANGING	0.0	
BCS_INCREASING	1.0	
BCS_DECREASING	(-1.0)	
BCS_CHANGE_MASK	3	manual: the bcs changes every 4 frames
HIGH_BCS_CHANGE_THRESHOLD	76	
HIGH_BCS_CHANGE	60.0	
LOW_BCS_CHANGE	10.0	
MAX_BCS_SETTING	4000.0	
MIN_BCS_SETTING	200.0	
DEFAULT_BCS_RANGE	1200.0	
BCS_BOOTUP_TIME	2	seconds
NUM_SLEW_ENTRIES	15	ticks (1.0 sec)

int declarations:

range_entered_manually	TRUE or FALSE
bcs_manual_range_change	0 if bcs range not changing, 1 if bcs range is increasing, -1 if bcs range is decreasing time the bcs has been changing
bcs_change_elapsed	
bcs_bootup_timer	
bcs_booted_up	TRUE or FALSE
flashing_zeros	
slew_rate_ptr	

REAL declarations:

bcs_range	in meters
super_elevation	in radians
lead_azimuth	in radians
gps_slew_rate	in radians/sec
flight_time	in seconds
apds_yb[10]	arrays to store genbal data
apds_zb[10]	
heat_yb[10]	
heat_zb[10]	
slew_rate_buffer[NUM_SLEW_ENTRIES]	

ObjectType declarations:

ammo_type_selected

char declarations:

bcs_range_str[80]

Pointer to char declarations and initialization:

bcs_range_format = "%04d"

2.2.3.1.1 bcs_dump_lead_buffer

The slew rate buffer is flushed and the slew rate pointer is initialized.

2.2.3.1.2 bcs_init

This routine initializes the ballistics computer. The initial bcs range is set to 1200 by default. The parameter files are loaded.

Calls	
Function	Where Described
bcs_dump_lead_buffer	Section 2.2.3.1.1
timers_set_null_timer	Section 2.6.3.14.1
ballistics_load_parameter_file	Section 2.5.2.3.3

Table 2.2-58: bcs_init Information.

2.2.3.1.3 bcs_simul

This routine simulates the bcs system. It controls the operation of the manual battlesight. When the manual bcs button is pushed, the bcs range is set to the default range(1200 meters). Pushing the add(drop) switch causes the bcs range to be increased (decreased) by ten meters. If this switch is held down, the range will be increased (decreased) nineteen times over the next five seconds. The increase (decrease) is in increments of ten meters. If the switch continues to be held down after the initial five seconds, the range changes at the same rate but in increments of sixty meters. This change will continue until the range reaches the maximum (minimum) value of 4000 meters (200 meters).

The process is modelled in roughly the same manner, however, the events are timed in frames rather than seconds. The range changes once every four frames rather than 19 times in five seconds.

Internal Variables		
Internal Variable	Type	Where Typedef Declared
abs bcs change	register REAL	sim_types.h
b range	pointer to register REAL	sim_types.h
Calls		
Function	Where Described	
bcs_check_bootup	Section 2.2.3.1.23	
max	sim_macros.h	
min	sim_macros.h	
turret_get_gps_slew_rate	Section 2.2.6.1.1.4	

Table 2.2-59: bcs_simul Information.

2.2.3.1.4 bcs_manual_range_battlesight

This routine is called when the Battle Sight button is pushed. It sets the bcs range status to manual.

2.2.3.1.5 bcs_manual_range_add_pushed

This routine is called when the Add switch is depressed. This is not a momentary depression; the switch must be actively released. This routine increases the range manually.

2.2.3.1.6 bcs_manual_range_drop_pushed

This routine is called when the Drop switch is depressed. This is not a momentary depression; the switch must be actively released. This routine decreases the range manually.

2.2.3.1.7 bcs_manual_range_released

This routine is called when the Add (Drop) switch is released. It stops the range from changing further.

2.2.3.1.8 bcs_ammo_select_heat

This routine sets the ammo type when heat is selected. It sets ammo_type_selected to munition_US_M456A1.

2.2.3.1.9 bcs_ammo_select_apds

This routine sets the ammo type when apds is selected. It sets ammo_type_selected to munition_US_M392A2.

2.2.3.1.10 bcs_ammo_select_other

This routine sets the ammo type when other is selected. It sets ammo_type_selected to munition_US_M392A2.

2.2.3.1.11 bcs_range_is

This routine is called to clamp the new range whenever the laser range finder determines a new value.

Parameters		
Parameter	Type	Where Typedef Declared
range_from_lrf	REAL	sim_types.h
Internal Variables		
Internal Variable	Type	Where Typedef Declared
b_range	pointer to register REAL	sim_types.h

Table 2.2-60: bcs_range_is Information.

2.2.3.1.12 calc_avg_slew_rate

This routine returns the average slew rate.

Internal Variables		
Internal Variable	Type	Where Typedef Declared
i	int	Standard
slew_rate_sum	REAL	sim_types.h
Return Values		
Return Value	Type	Meaning
slew_rate_sum/NUM_SLEW_ENTRIES	REAL	The average slew rate

Table 2.2-61: calc_avg_slew_rate Information.

2.2.3.1.13 bcs_set_ballistics_computer

This routine performs simulation of the on-board ballistics computer. It calculates actual flight time and superelevation of any projectile given the bcs_range. The lead_azimuth for lead tracking is also calculated.

Errors	
Error Name	Reason for Error
BCS: Unknown ammo type ... selected	The variable ammo_type_selected is not one of the expected values.
PANIC(?) -- bcs time == 0.0	Projectile flight time is zero.
Calls	
Function	Where Described
ballistics_calc_time	Section 2.5.2.1.1
ballistics_calc_se	Section 2.5.2.1.2
eq	tolerance.h
calc_avg_slew_rate	Section 2.2.3.1.12

Table 2.2-62: bcs_set_ballistics_computer Information.

2.2.3.1.14 bcs_get_lead_azimuth

This routine returns the lead azimuth.

Return Values		
Return Value	Type	Meaning
lead_azimuth	REAL	The lead azimuth

Table 2.2-63: bcs_get_lead_azimuth Information.

2.2.3.1.15 bcs_get_super_elevation

This routine returns the super elevation.

Return Values		
Return Value	Type	Meaning
super_elevation	REAL	The super elevation

Table 2.2-64: bcs_get_super_elevation Information.

2.2.3.1.16 bcs_get_range

This routine returns the ballistics computer system range.

Return Values		
Return Value	Type	Meaning
bcs_range	REAL	The bcs range

Table 2.2-65: bcs_get_range Information.

2.2.3.1.17 bcs_get_time_of_flight

This routine returns the time of flight of the projectile.

Return Values		
Return Value	Type	Meaning
flight time	REAL	The time of flight

Table 2.2-66: bcs_get_time_of_flight Information.

2.2.3.1.18 bcs_get_ammo_type_indexed

This routine returns the type of ammunition selected.

Return Values		
Return Value	Type	Meaning
ammo_type_selected	ObjectType	The selected ammunition type

Table 2.2-67: bcs_get_ammo_type_indexed Information.

2.2.3.1.19 bcs_get_range_str

This routine is called by the graphics interface to get the range to be displayed on the screen of the gunner's view. The range is an integer rounded to the nearest 10 meters.

Internal Variables		
Internal Variable	Type	Where Typedef Declared
flash cnt	int	Standard
Return Values		
Return Value	Type	Meaning
bcs_range_str	pointer to char	The bcs range string

Table 2.2-68: bcs_get_range_str Information.

2.2.3 : 20 bcs_boot_computer

This routine is called by controls to boot the ballistics computer system when turret power is turned on.

Calls	
Function	Where Described
timers_get timer	Section 2.6.3.6.1

Table 2.2-69: bcs_boot_computer Information.

2.2.3.1.21 bcs_computer_status

This routine returns the computer status.

Return Values		
Return Value	Type	Meaning
BCS BOOTED UP	int	bcs booted up successfully.
BCS STILL BOOTING	int	bcs boot in process.
BCS OFF	int	bcs has not booted up.
Calls		
Function	Where Described	
timers_get in use status	Section 2.6.3.7.1	

Table 2.2-70: bcs_computer_status Information.

2.2.3.1.22 bcs_turn_computer_off

This routine turns off the ballistics computer system. This routine is called when turret power is turned off.

Calls	
Function	Where Described
timers_set null timer	Section 2.6.3.14.1
timers_free timer	Section 2.6.3.5.1

Table 2.2-71: bcs_turn_computer_off Information.

2.2.3.1.23 bcs_check_bootup

This routine checks the ballistics computer system bootup.

Calls	
Function	Where Described
timers get in use status	Section 2.6.3.7.1
timers get ticking status	Section 2.6.3.20.1
timers free timer	Section 2.6.3.5.1
timers set null timer	Section 2.6.3.14.1

Table 2.2-72: bcs_check_bootup Information.